NAVAL POSTGRADUATE SCHOOL
MONTEREY, CALIFORNIA

THESIS

OPTIMIZATION METHODS FOR MIXED MINEFIELD CLEARANCE

by

David D. Romberger

September 1996

Thesis Advisor: Alan R. Washburn

Approved for public release; distribution is unlimited.

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	September 1996	Master's Thesis

4. TITLE AND SUBTITLE	5. FUNDING NUMBERS
OPTIMIZATION METHODS FOR MIXED MINEFIELD CLEARANCE	
6. AUTHOR(S) David D. Romberger	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey CA 93943-5000	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION/AVAILABILITY STATEMENT	12b. DISTRIBUTION CODE
Approved for public release; distribution is unlimited.	

13. ABSTRACT *(maximum 200 words)*

This thesis describes the development and implementation of an improved optimization feature for the minefield clearance TDA MIXER. A constrained form of MIXER's original local optimal search method is proposed, followed by an exhaustive search method, and then a simulated annealing method.

Computational efficiency and program run times are examined for the exhaustive search method. Also, a performance comparison of "optimal" solutions for the local search and simulated annealing methods is given. A final version of the optimization feature incorporates all three search methods.

14. SUBJECT TERMS	15. NUMBER OF PAGES
mine warfare, TDA, mine clearance optimization, simulated annealing, FORTRAN	85
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18 298-102

Approved for public release; distribution is unlimited.

OPTIMIZATION METHODS FOR MIXED MINEFIELD CLEARANCE

David D. Romberger
Lieutenant , United States Navy
B.S.N.A., United States Naval Academy, 1988

Submitted in partial fulfillment
of the requirements for the degree of

MASTER OF SCIENCE IN OPERATIONS RESEARCH

from the

NAVAL POSTGRADUATE SCHOOL
September 1996

ABSTRACT

This thesis describes the development and implementation of an improved optimization feature for the minefield clearance TDA MIXER. A constrained form of MIXER's original local optimal search method is proposed, followed by an exhaustive search method, and then a simulated annealing method.

Computational efficiency and program run times are examined for the exhaustive search method. Also, a performance comparison of "optimal" solutions for the local search and simulated annealing methods is given. A final version of the optimization feature incorporates all three search methods.

TABLE OF CONTENTS

LIST OF FIGURES

I. INTRODUCTION

This thesis proposes and tests improvements to the optimization feature in the tactical decision aid (TDA) MIXER [Ref. 1]. MIXER is a FORTRAN program intended as a tool to be used during naval mine clearance operations. This thesis begins with a brief discussion of the mine threat problem and the Navy's current best software tools designed to deal with mines. MIXER is then introduced with a short description of its features not concerned with optimization. Next MIXER's optimization feature is presented with a detailed break down of the underlying theory and assumptions it uses to model minesweeper-mine interactions. Then a description of the present optimization method and of some associated shortfalls is given. Next several approaches to improve the optimization method are proposed, implemented, and tested. Finally a brief discussion of areas open for further analysis is presented.

A. NAVAL MINE WARFARE

Naval mine warfare is laying or clearing mines to prevent or restore the use of sea ways, harbors and coastal waters. Examples include clearing boat-lanes for amphibious landings, clearing channels or shipping lanes to allow free movement of commercial traffic, or laying mines in coastal waters and ports to deny access to enemy forces. The U.S. Navy's mine warfare community is charged with accomplishing these tasks. The various types, abundance, and relatively low cost of mines make mine clearance a very difficult problem to solve.

Mines today range from bulky WWII mines to new sleek Manta mines. Mines can actuate by pressure, magnetic, and/or acoustic influence, as well as by contact. Mines are easily deployed from most seagoing vessels, and have been used successfully in many conflicts throughout history. Today, whether old or new, simple or sophisticated, a minefield is

still a potential "show stopper".

Tools for mine clearance are as varied as the mines themselves. Special ships with quiet propulsion and low magnetic signatures enter minefields and locate mines with sonar. Helicopters pull sleds capable of exposing and actuating mines. Teams of divers and sea mammals are trained to locate and destroy mines. Other tools include airborne mounted laser systems that can detect mines several meters below the sea surface, and nets of exploding line charges that are hurled across the surf zone to the beach. These and other systems are used separately or in combination to neutralize minefields.

The Navy has a history of fluctuating emphasis with regard to mine warfare. Resources and R&D efforts tend to peak and wane with the latest mine incident. Recently, however, the Navy appears ready to cease past trends, with the procurement of a new and capable Mine Countermeasure (MCM) force. The growing base of assets in Corpus Christi, TX, and the newly converted MCM Command and Control ship (USS INCHON) point to a strong commitment. Although mine and mine clearance technology progresses, many current assets remain untested in a real war environment.

1. Are sea mines still a threat?

Worded differently the question is whether or not the Navy really needs a strong MCM force? The USS TRIPOLI and USS PRINCETON incidents during the Persian Gulf crisis indicate there was a need then. Those encounters with mines were nearly six years ago - what about today? There are signs today that may indicate a greater need then ever.

Mines are more abundant today, as many countries manufacture mines for sale on the open market. The frequency of small regional conflicts in which poorer third world

countries are involved is on the rise. Those countries with larger coastlines and small pocketbooks may look to the inexpensive sea mine as an effective way to prevent interference from the sea.

In an era when national interests turn Naval operations towards littoral waters, mines are a huge concern. The late Chief of Naval Operations, Admiral Jeremey Boorda recently explained, "With 95% of all materials to be sent to support future regional conflicts going by sea, the ability to close vital waterways comprises a threat of strategic dimensions." [Ref. 2]

The answer to the question posed in the heading then is "Yes", mines are still a threat, and the Navy needs a strong countermeasure. Though historically a less emphasized issue for the Navy, today development of new equipment and resources for its MCM forces is a top priority.

2. Goal of the Mine Warfare TDA

In the mine laying and mine countermeasure arena, the Navy has many assets at its disposal like the ones mentioned earlier. To complement these physical tools, the Navy is using computers to improve its mine warfare capabilities. The Navy's C4I architecture incorporates computer assisted data and communication links as key elements in maintaining an accurate and up-to-date "big picture". As a genuine warfare area, Mine Warfare has its niche within this architecture. At the heart of this niche lies the tactical decision aid.

The goal of the mine warfare TDA is to help plan, practice, evaluate and conduct mine laying and clearance operations. The TDA gives the mine warfare commander a tool to develop a plan of action, or to warn against a possibly bad plan of action. A good TDA accomplishes this task in an easy and timely fashion.

The mine warfare TDA, like any simulated or analytical analysis tool, is no better then the information it is given. Mine warfare, by design, possesses many unknowns, and clearance operations are usually information starved. During MCM missions unknowns such as the numbers and types of mines present, the boundaries of the minefield, the most effective way to employ mine warfare assets, and even the effectiveness of those assets are difficult to determine. Intelligence, surveillance and reconnaissance may give some insight to these questions; however, the problem is still difficult.

The mine warfare model is a simulated or analytic approach to minefield construction or countermeasure. An analytic model limits the definition and scope of the unknowns involved to a point where proven relations and theoretical equations can be employed. Analysis of this type tends to be less time consuming, but also requires many simplifying assumptions. Computer simulation, on the other hand, allows greater freedom for the unknowns, achieving accuracy through extensive replication. Although more realistic, large simulated models are slow and not well suited for tactical purposes. A good TDA attempts to incorporate the best of both worlds, providing sufficient realism and speed to be accurate and usable on existing computers.

3. Current Mine Warfare Analysis Tools

A model must start with some basic assumptions. Mines are tricky, but so far none have been developed to detonate twice. Therefore all models start with this assumption. Although one mine's detonation may actuate other mines nearby, or possibly trigger a timing mechanism for future actuations, all TDA's at present assume no mine interaction. The data needed to model this would be nearly impossible to derive. Mines and minefields can be very sophisticated and are rarely

predictable.

Some theater level models, such as the Navy's wargame model ENWGS, include minefield threats. Although lacking the detail needed for a TDA, ENWGS does make use of some basic mine warfare modeling techniques. Ships entering a minefield are destroyed with some probability based on basic geometric relationships between the area of the minefield, the distance the ship has traveled in the minefield and the actuation radius of the mine [Ref. 3: p. 5]. When the mine actuates, the mine and ship (unless designated an MCM unit) are both removed from the model. In this way ENWGS allows the users to experiment with minefield location, account for assets needed to lay and remove mines, and observe the effects minefields have on the overall battle. ENWGS is a simulation model and minefields are only one of many details it must consider.

Most tactical level models are designed for either minefield planning or mine countermeasures. The Navy's Uncountered Minefield Planning Model (UMPM) is an analytic model used for tactical minefield planning. UMPM employs damage curves; the probability of a ship being destroyed by an actuated mine depends on the separation distance at detonation. UMPM outputs several Measures of Effectiveness (MOE's) which describe or quantify the "goodness" of a particular plan [Ref. 3: p. 10]. One such measure is simple initial threat (SIT), which is the probability that the first unit to transit the minefield will be destroyed. This quantity will be examined more closely later. UMPM is a more detailed minefield model than ENWGS, and provides measures for closer analysis of minesweeper-mine interactions.

A major flaw in UMPM's model is its use of "pre-averaged" actuation probabilities [Ref. 3: p. 13], which do not allow separate calculations for each type of mine encountered. The result in certain situations is an over estimation in the

ability of the minefield to kill ships. This would clearly be an unsafe error from the minefield planner's perspective.

There are several examples of TDA's designed for mine countermeasures, including the Navy's Non Uniform Coverage Evaluator (NUCEVAL) and Uniform Coverage Planner (UCPLAN). Unlike a minefield planning model, these models attempt to distribute rather than channelize the tracks (mine sweepers) in the minefield. The objective of these models is to resolve the fraction of mines removed from the minefield following execution of a particular sweep plan. NUCEVAL asks the user to input a sweep plan and then provides the fraction of mines swept, while UCPLAN asks for the desired clearance level and produces a sweep plan. Minesweeper-mine interactions proceed in a similar fashion described for UMPM.[Ref. 3: p. 16]

In assessing the effectiveness of a particular sweep plan the results provided by NUCEVAL and UCPLAN are incomplete at best. These models fail to address several very important factors relevant to most tactical clearance operations; the number of mines present, the types of mines present, and the possibility of damage to MCM assets.

A third mine clearance TDA, Cognitive Planning Aid (COGNIT), addresses the number of mines and asset damage possibility. COGNIT is an analytic TDA, written in FORTRAN in the late 1980's, designed to assist the MCM team in developing an "optimum tactic" or sweep plan [Ref. 4]. In addition to damage probabilities and navigation errors, COGNIT's minefield model includes the effects of mine counter-countermeasures. Each mine has an assigned ship count setting which allows mines to actuate several times prior to detonation.

COGNIT incorporates the number of mines present by use of a cross channel mine density (mines/nm), which is constant over the width the minefield. The number of mines swept is determined by the number of detonations that occur during each

minesweeper's runs or tracks through the minefield.

COGNIT incorporates three problem types. The problem type is selected by the user upon execution of the program. Each problem type corresponds to a different measure of effectiveness (MOE). The three MOE's are countermeasure effort (accounts for sweep time), average clearance level, and expected number of casualties to countermeasure platforms. COGNIT treats one of the MOE's as the primary objective to be either maximized or minimized, and the remaining two MOE's as the constraints. The user proceeds by running each problem type several times, making manual trade-offs among the constraints between each run, until a plan is produced that adequately satisfies each MOE.

Although much improved over previous TDA examples, COGNIT does have several drawbacks. Although mines in the minefield may be given different ship count settings, all mines must be the same type. In addition, all sweepers must be the same type. COGNIT can not model a situation involving different mine and sweep types simultaneously. COGNIT's optimization method is iterative in nature requiring the user to make decisions and trade-offs between each iteration.

B. WHY THIS THESIS?

The new prototype TDA MIXER, [Ref. 1], attempts to correct many of the shortfalls of current mine clearance models. MIXER provides a combination of simulation and analytic tools for analysis of minefield clearance plans. MIXER can evaluate a given plan or produce an optimal plan given the necessary data. MIXER is still in its infancy and is presently not very user friendly, but it does provide the foundation for further development. The purpose of this thesis is to improve MIXER's optimization feature.

1. Main Purpose

MIXER includes an optimization subroutine OPT that provides the user a mine clearance (or sweep) plan. Optimization only occurs at the request of the user. This feature minimizes a "cost" that accounts for the reduction in SIT as a result of sweeping, for the potential danger to the sweeping units, and for sweep time. OPT requires quantifying the value of mine sweeper assets and gives full consideration to the vulnerability of those assets as they clear mines.

Accounting for the cost for each sweep plan requires OPT to solve a combinatorial optimization problem that is nonconvex and integer in nature. The problem is combinatorial because the objective function is a collection of terms that are not analytically dependent or related. The problem is integer because the solution is based on the premise that once the decision is made to continue sweeping, at least one complete transit (or sweep) through the minefield with a chosen sweeper must occur. The nonconvexity of the problem and detailed descriptions of all terms will be discussed at length in the following chapters.

OPT finds a solution that is locally optimal. Given an initial sweep plan, OPT finds a better plan as long as it can be obtained by incremental changes without an increase in cost. This solution would be the absolute best, or globally optimal, if the objective function were noninterger and convex. With the current objective function, however, all that can be said is a locally optimal solution may be better than the initial solution. Clearly if improvements to the locally optimal plan can be realized then it is worth some investigation.

Another aspect of OPT that has area for improvement is its consideration of sweep time, which is the total time mine sweepers spend hunting and/or sweeping in the minefield. OPT

includes sweep time as a cost term in the objective function. The sweep time is multiplied by a "value" for time and added to other cost terms. The value factor acts as a Lagrangian multiplier that relaxes an otherwise constrained version of the problem. If the mine warfare commander can estimate the "value" for sweep time then he can use OPT to get a locally optimal sweep plan. The actual sweep time is left for OPT to compute. Although this method of handling sweep time removes the constraint, a commander pressed for time himself would most likely prefer a direct constraint. If a time constraint does not extensively complicate an already difficult problem, then its inclusion is also worth some investigation.

It is important here to discuss the actual run time required for MIXER to perform optimization, especially considering the above complications. OPT finds the locally optimal sweep plan almost instantaneously. A tactical decision aid, emphasis on tactical, could be considered deficient if it required extensive run time to solve its problem. Therefore, any change to OPT should not be considered an improvement unless the additional run time required is minimal.

2. Overview of Improvement Process

The first step in solving any problem is gaining a firm understanding of the problem itself. Chapter II describes MIXER and gives a detailed description of OPT in its present form. In Chapter III the objective function in OPT is evaluated, its terms defined, and its computations explained. Then various changes to MIXER's objective function are presented. In Chapter IV several optimization methods are applied and tested. Then test results are examined to determine if any improvement occurred. Finally, in Chapter V a review of the thesis is given followed by suggestions for further research.

II. MIXER: A TDA FOR MIXED MINEFIELD CLEARANCE

MIXER is a menu driven computer program written in FORTRAN. MIXER gets its information from pre-constructed data files (see Appendix A and B), as well as from questions it asks the user during execution. MIXER outputs results in numeric tables to the screen and to a data file.

A. PROGRAM EXECUTION

MIXER consists of a main program and several subroutines each with a specific task. The main program (MAIN) supervises the overall execution of MIXER, and directs the user to the desired application. MAIN reads data from the data files and writes results to data files, and to the screen. MAIN terminates execution once the user is finished. The original version of MAIN can be found in [Ref. 1].

MAIN reads data from two files. The first data file is called PARAMS.DAT. This file contains most of the parameters that describe the minefield model and the minesweeper-mine interactions. Included in this file are the number of different mine, sweep and resource types. Resources are the individual assets that combine to form a sweep type. One "helicopter" (a resource) and one "sled" (another resource) might be combined to form one "magnetic sweep" sweep type. PARAMS.DAT contains an assignment table that allocates the number of resources required to form a specific sweep type. PARAMS.DAT also contains assignment tables for minesweeper-mine interactions. These include actuation distances or sweep fronts, actuation probabilities, danger distances or dangerous fronts, and damage probabilities. The actuation tables provide the distances and probabilities with which a particular sweep type can detonate a particular mine type. The danger and damage tables describe the distances and probabilities

required for a detonated mine to damage a particular sweep type.

Other information found in PARAMS.DAT include minefield dimensions, navigation errors, sweep speeds, resource and target traffic (here called high value unit) "values", and mine probability actuator settings. Probability actuators are mine counter-countermeasures which would be set by minefield planners to prevent mines from being swept. PARAMS.DAT was designed to hold information that would most likely be known prior to the start of MCM operations and that would not change as operations are carried out.

The second data file read by MAIN is NUMBERS.DAT. This file contains most of the information regarding the initial sweep plan. NUMBERS.DAT contains the number of tracks through the minefield each sweep type will use and the number of runs each sweep type will make on its respective tracks. It also contains the track positions measured in yards from the left side of the minefield. NUMBERS.DAT also holds information about the numbers of mines for each mine type suspected to be in the minefield, and the number of assets available for each resource type. The data in this file is intended as an initial guess or starting point from which MIXER can begin execution. Once MIXER completes an application that updates the initial plan, MAIN writes the new plan to a file called NEWNUM.DAT, which can then be renamed NUMBERS.DAT and used as the new initial plan for subsequent executions of MIXER.

Once MAIN has read the data files, it does preliminary actuation and damage threat calculations and performs a theoretical assessment of the given initial sweep plan. MAIN then outputs to the screen a tabular description of the sweep plan, the clearance levels for each mine type, the sweep time required, an approximate SIT, and the average number of resources lost during sweeping. MAIN then asks the user to

select one of its primary applications from MIXER's main menu. Once MAIN reads the input from the user it executes the chosen application and passes control to the respective subroutine. When an application has completed execution the user can either run the application again or return to the main menu and select another application, or terminate the program. Upon termination MAIN writes the final sweep plan to NEWNUM.DAT.

There are five different applications that can be selected from MIXER's main menu. First is the optimization feature (OPT) which will be discussed at length later. The second application (INPUT) allows the user to make manual changes to the initial sweep plan by selecting a desired sweep type and changing the total number of runs. The third application is MIXER's Monte Carlo simulation feature (MONTE). MONTE gives a table of results that includes the percentage of each sweeper's runs completed, the average number of mines from each mine type that were swept, the average loss of resource types, and SIT. The accuracy of MONTE's calculations are dependent on the number of replications, but MONTE is potentially the most accurate application in MIXER for determining the effects of a particular sweep plan.

The fourth application (REHEARSE) runs a single replication of the simulation in MONTE. REHEARSE outputs the remaining mine and resource data to NEWNUM.DAT, which can then be used as the initial numbers, along with a new initial sweep plan, in the event further sweeping is desired. With this application the MCM operations can be tested in phases. As each phase of the operation completes the final mine and resource numbers are saved as the starting point for the next phase.

The fifth and final application is REALITY. It is similar to REHEARSE except that instead of running a simulation, actual results from an ongoing MCM operation are used. As

mines and resources are actually depleted their numbers are entered in REALITY. FORTRAN code for all applications is given in [Ref. 1].

B. OPTIMIZATION

The remainder of this thesis is dedicated to MIXER's optimization application (OPT). As was mentioned in Chapter I, the original version of OPT finds the local best solution to a combinatorial nonconvex integer problem. The details of the problem will be given in Chapter III. In this section the general execution of OPT will be explained, and some background information regarding OPT's methodology and assumptions will be given.

Once selected OPT asks the user to input a price for time (*LAMBDA*). The value input for *LAMBDA* is completely arbitrary except that a larger input value will produce a plan that requires less sweep time, while a smaller value will produce a plan that requires more sweep time. OPT then executes, finds its best sweep plan, and outputs a table of results. The table includes the sweep time, SIT, and the average losses to sweeper assets. Then OPT asks the user to input a new value for *LAMBDA* or to return to the main menu.

OPT provides the mine warfare commander a place to start preparing for mine clearance operations. Given the best guess information about the minefield and the MCM assets on hand, OPT produce's a sweep plan that theoretically can be executed in a set amount of time and produce results that are in some sense optimal. The decisions OPT makes are based on theoretical calculations and analytic relationship's that are only rough approximations of the real world. OPT does, however, attempt to account for the value of the sweep resources and of the high value unit (HVU), and of course MIXER's Monte Carlo simulator (MONTE) can be used to test

OPT's results for a more accurate indication of its effects on the minefield.

1. The Minefield Model and Lost Resources

The minefield model MIXER uses for optimization is defined by the data provided in PARAMS.DAT and NUMBERS.DAT. In this model the minefield is treated as a single entity that has varying states. In each state the minefield responds differently depending on the type of stimuli it is presented. The states of the minefield correspond to the level of clearance that exists at any given time. The stimuli are the different sweep types and the HVU that pass through the minefield.

The sequence in which sweep types pass through the minefield is one of the inputs in PARAMS.DAT. A sweep type does not begin its runs through the minefield until the previous sweep type is finished, the idea being to prevent detonations by one sweep type from damaging another. It might be expected that the least vulnerable MCM assets would be used first, however, this is not a requirement, nor is it assumed. The only requirement with regards to the sequence order is that the HVU is last, for obvious reasons.

The modeled interaction between the mines and sweep types is defined by two important derived arrays, $SURV(I,J)$ and $TH(I,J)$. Their values are calculated by MAIN with data from the actuation and damage threat tables provided in PARAMS.DAT. The equation for $SURV(I,J)$ is

$$SURV(I,J) = 1 - \frac{A(I,J) \times B(I,J)}{WIDTH_{MINEFIELD}} \quad , \tag{1}$$

where $A(I,J)$ is the actuation width for sweep type J confronting mine type I, and $B(I,J)$ is the probability that

sweep type J actuates mine type I, given mine I is within sweeping range of sweep J. $SURV(I,J)$ is, therefore, the probability that a mine of type I remains in the minefield following one complete run of sweep type J.

The basic equation for $TH(I,J)$ is

$$TH(I,J) = \frac{AF(I,J) \times BF(I,J)}{A(I,J)} \quad , \tag{2}$$

where $AF(I,J)$ is the distance required for mine type I to damage sweep type J, and $BF(I,J)$ is the probability that a mine of type I will damage sweep type J. $BF(I,J)$ is conditional on mine type I actuating, and on sweep type J closing to the damage distance. $TH(I,J)$ is, therefore, the probability that mine I kills sweep type J, given mine I is actuated by sweep type J. The actual computation of $TH(I,J)$ includes a trapezoidal factor which tends to sheer off the corners of an otherwise square actuation curve [Ref. 3: p. 6]. Unlike MONTE, which assigns sweep tracks in accordance with the current sweep plan, OPT assumes tracks are located independently across the width of the minefield.

As sweeping progresses the mean number of mines remaining decreases. In subroutine F OPT calcultes the probability that any mine of type I remains. Values are held in $Q(I)$, which is initially set to one for all mine types. As each sweep type completes its turn in the minefield, $Q(I)$ is reduced accordingly. The equation for $Q(I)$ is

$$Q(I) = \prod_{J=FIRST\ IN\ SEQ}^{J=PRIOR\ IN\ SEQ} SURV(I,J)^{X(J)} \quad , \tag{3}$$

where $X(J)$ is the total number of runs sweep type J makes during its turn, a decision variable in OPT. At the start of any given sweep type's turn, $Q(I)$ is the probability that a

mine of type I still exists. Upon completion of the turn $Q(I)$ is updated for the next sweep type. This is how the the levels of clearance for the minefield model change.

Equations 1, 2 and 3 fully describe the minesweeper-mine interaction OPT employs. These terms can now be combined to compute the probability of actuating a mine, and the probability of damaging a sweep type, at any given point in the sweep plan. The following relationships restate the above terms;

$$Q(I)=PROB\{ANY\ MINE\ OF\ TYPE\ I\ EXISTS\}, \tag{4}$$

$$SURV(I,J)=PROB\{J\ DOES\ NOT\ ACTUATE\ I \mid I\ EXISTS\}, \tag{5}$$

$$TH(I,J)=PROB\{I\ KILLS\ J \mid J\ ACTUATES\ I\}. \tag{6}$$

The probability of a given mine of type I being actuated by some minesweeper of type J is given by $Q(I)\times[1-SURV(I,J)^{X(J)}]$, and the probability that a mine kills a sweeper is given by $TH(I,J)\times Q(I)\times[1-SURV(I,J)^{X(J)}]$.

The above minefield model provides the basic theory behind each mine encounter. Aside from a few further assumptions, [Ref. 1: p. 7], this is the foundation from which OPT computes the changing cost in the objective function. The cost term that accounts for the losses to resources is based on the probability that a mine kills a sweeper, $TH(I,J)\times Q(I)\times[1-SURV(I,J)^{X(J)}]$. Combining this term for each mine with the values for each resource type gives the cost of losses to the MCM force for the proposed sweep plan [Ref. 1: p. 9]. $LOSS(K)$, where K is the resource type, is the average number of resources lost due to damage by mines. This is the first of the three terms in the objective function and from

here on is referred to as "resource cost". A small number for resource cost should indicate that the associated sweep plan is effective against mines, but not excessively wasteful to the mine sweeps.

2. The KATZ Distribution and SIT

The HVU only makes one run through the minefield, and SIT is the probability that it is destroyed during that run. SIT is computed in OPT using a KATZ distribution. This class of distributions is particularly useful for estimating the number of mines in a shrinking minefield. KATZ distributions have two parameters, which are similar to mean and standard deviation in the normal distribution, and are analytically derived from them. The mean number of mines for each mine type and their respective standard deviations are given in NUMBERS.DAT. When MIXER is executed, MAIN converts mine data into the corresponding KATZ parameters. A detailed explanation on the benefits of using the KATZ distribution is described in [Ref. 7]. One benefit is a simple formula for SIT, namely

$$SIT = 1 - \prod_{ALL\ MINE\ TYPES} \left[\frac{1 + KZB(I) \times TH(I, JMAX) \times Q(I)}{1 - KZB(I)} \right]^{-KZA(I)/KZB(I)}, \quad (7)$$

where $KZA(I)$ and $KZB(I)$ are the KATZ parameters for mine type I [Ref. 3: p. 18]. Subroutine F does the actual calculations. SIT is then combined with the value of the HVU as the second cost term, from here on referred to as "SIT cost", in the objective function. This SIT cost accounts for the sweep plans effectiveness against the minefield in terms of safety to future traffic. A small number for SIT cost should be a good indication that most if not all of the of the mines have been cleared.

3. The Lagrangian Multiplier and Sweep Time

The third and final cost term in MIXER's objective function accounts for the sweep time required to execute the current sweep plan. Sweep time is measured in hours and is a computed by adding the hours each sweep type requires to complete its turn in the minefield. The hours required to complete one run of a particular sweep type depends on its speed, turn velocity, and on the length of the minefield [Ref. 1: p. 6]. This hours-per-run term is then multiplied by the number of runs for each sweep type, and then summed over all sweep types to get the total sweep time.

OPT calculates sweep time for the proposed sweep plan and then multiplies the sweep time by the user's input for the Lagrangian multiplier *LAMBDA* to get "sweep time cost". Sweep time cost is then added to the resource cost and SIT cost to arrive at a total cost for the current plan. This Lagrangian method allows OPT to execute in an unconstrained mode. To summarize, the objective function in OPT is

minimize: (8)

$$MOE = LAMBDA \times TIME + \sum_{K=1}^{KMAX} VAL(K) \times LOSS(K) + VAL(KMAX+1) \times SIT \ ,$$

where $VAL(K)$ is the value given to the Kth resource type, and $KMAX+1$ is the resource index of the HVU [Ref. 1: p. 8].

4. Unconstrained Local Optimization

OPT changes the total cost of a particular sweep plan by changing the total number of runs of a particular sweep type, $X(J)$, with all other terms held constant. It is important to note that $X(J)$ is actually a combination of two separate terms. The equation for $X(J)$ is

$$X(J) = TOTRUN(J) \times NP(J) \quad , \tag{9}$$

where $TOTRUN(J)$ is the actual number of runs a sweep type makes, and $NP(J)$ is the number of resource dependent parallel sweeping units that make up sweep type J. $NP(J)$ is a fixed value completely dependent on the number of resources available. It is assigned its values in MAIN. $TOTRUN(J)$ is variable, and the only factor that OPT controls. OPT finds an optimal sweep plan by determining the optimal runs, $TOTRUN(J)$, for each sweep type. For the rest of the thesis $TOTRUN(J)$ will be used when referring to sweep runs.

When OPT is executed it begins by computing the total cost (Equation 8) for the current sweep plan. OPT then attempts to increase or decrease $TOTRUN(J)$ for each sweep type by considering only unit changes, and compares the new total cost with the old total cost. If a complete loop through each sweep type produces no improvements in total cost, OPT terminates, saving the sweep plan $TOTRUN(J)$ corresponding to the lowest total cost. The pseudo code in Figure 1 outlines OPT's local optimization algorithm.

```
1. Get initial sweep plan TOTRUN(J).
2. Set best = cost(TOTRUN(J)).
3. Set J equal to first sweep type.
   3.1 Continue until J equals last sweep type.
      3.1.1 Set TOTRUN(J)' = TOTRUN(J) + 1.
      3.1.2 If cost (TOTRUN(J)') < best,
            best = cost (TOTRUN(J)'),
            TOTRUN(J) = TOTRUN(J)',
            Got to 3.
      3.1.3 If cost (TOTRUN(J)') > best and TOTRUN(J) > 0,
         3.1.3.1 Set TOTRUN(J)' = TOTRUN(J) - 1.
         3.1.3.2 If cost (TOTRUN(J)') < best,
                  best = cost (TOTRUN(J)'),
                  TOTRUN(J) = TOTRUN(J)',
                  Got to 3.
   3.2 Set J equal to next sweep type.
4. Return TOTRUN(J)
```

Figure 1. Local Optimization

5. Shortfalls

The two primary shortfalls in MIXER's current optimization feature are the optimization method itself, and the unconstrained, or Lagrangian method, for handling sweep time. The objective function, as mentioned earlier, is nonconvex and integer. This problem is a classic case that is difficult and well studied. A visualization of the objective function may provide some insight to the difficulties facing the local optimization method.

Three dimensional plots, Figures 2 and 3, can be obtained by limiting the modeled scenario to two sweep types. In this sparse scenario there are six mine types. Two of the mine types are very effective against the sweep types and ineffective against the HVU. Two other mine types are very effective against the HVU but ineffective against the sweep types. The remaining two mine types are moderately effective against both minesweepers and the HVU. The scenario is somewhat contrived in order to accentuate the nonconvexity of the objective function.

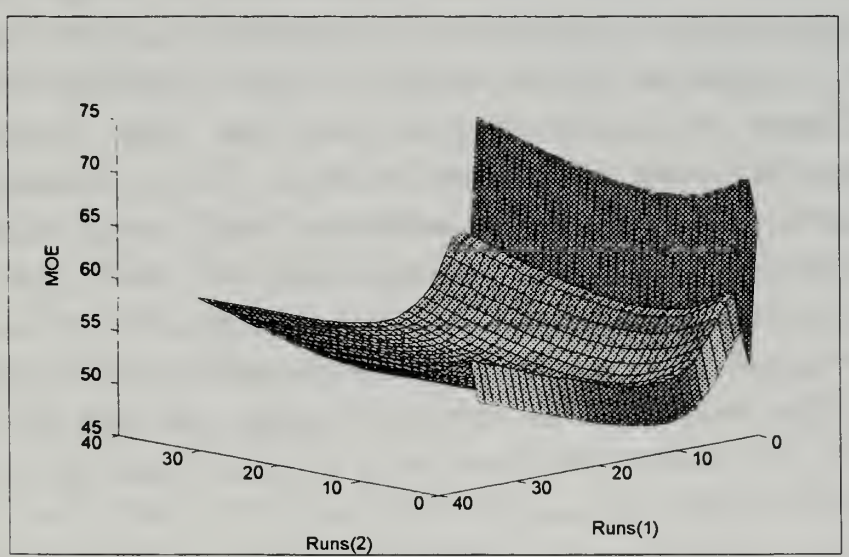

Figure 2. Surface Plot View 1

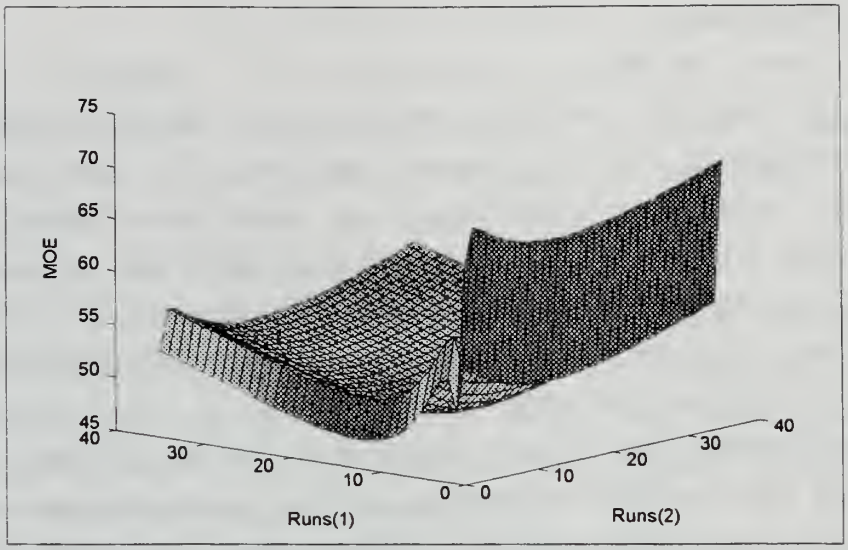

Figure 3. Surface Plot View 2

These figures show the number of runs for each sweep type plotted against the total cost of the MOE for *LAMBDA* equal to 2200. There are three local minimums. The first is located in the "valley" or center area, the second is along the Runs(1) axis, and the third is in a trough that parallels the Runs(2) axis. Since most of the surface area tends to slope into the center, this local minimum will probably be the favorite choice for a local optimizer, given a random initial solution.

In Figure 4 a contour plot for the same scenario is given. The staircase line leading from the top down to the "valley" area is the actual solution path OPT finds when presented this problem. As OPT updates its best plan it is actually moving on this path along the surface of the state space. The final plan at this local minimum has a total cost of 48.7. The solution at (0,12) along the Runs(1) axis, however, is the global minimum with a total cost of 47.7.

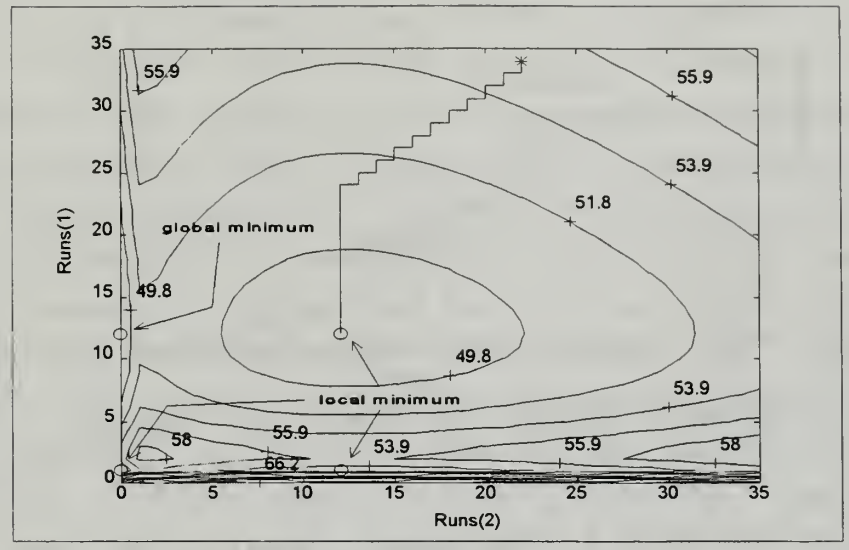

Figure 4. Path to Local Opt.

If it were possible to give an optimization method human characteristics, OPT would probably be called lazy and near sighted. OPT is lazy because it always selects the easy or downhill move. If OPT finds no downhill move it just gives up. OPT is nearsighted because it can only see one sweep run out in all directions. In all fairness, OPT does have some good qualities as well. OPT is fast, providing its answer almost instantaneously. OPT is also meticulous since its answer is always accurate. If OPT could be changed so that it could move uphill as well as downhill, and/or see over hills and around corners, then it would have a better chance of finding the global minimum. In this sense, OPT may just need a little training so that it won't be afraid to take on a few of those hills.

The second shortfall is the Lagrangian method OPT uses to handle sweep time. A plot of *LAMBDA* versus sweep time is given in Figure 5. These points were generated for the dense

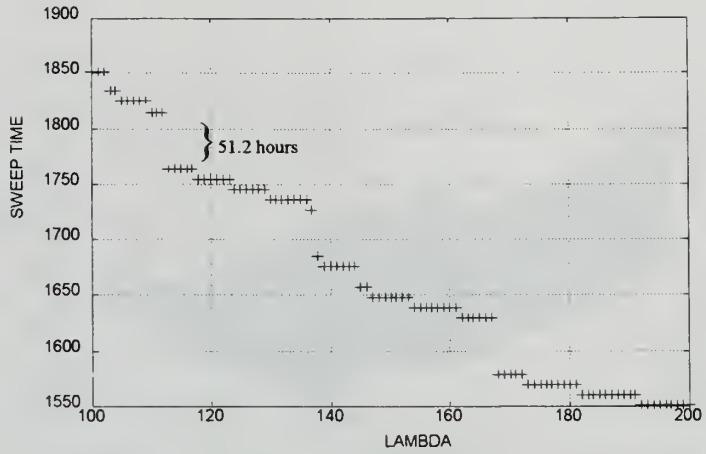

Figure 5. Sweep Time (hrs.) vs. *LAMBDA*

scenario in Appendix A. The dense scenario has seven sweep types and five mine types. The data in this scenario tends to be more realistic than in the sparse scenario.

Figure 5 reveals jumps between sweep times for changes in *LAMBDA*, and intervals of *LAMBDA* that produce levels of sweep time. Note that some of the jumps between the levels are rather large. The difference in sweep time for *LAMBDA* equal to 112 and 113, for example, is 51.2 hours, certainly enough time to complete additional sweep runs. Since each sweep time level corresponds to a particular sweep plan it appears that some sweep plans are being missed. In reality, if the mine warfare commander has an additional 51.2 hours he would be interested in a sweep plan that uses it. Unfortunately, with a Lagrangian method OPT can not locate such a sweep plan. A solution would be to move the Lagrangian term out of the objective function, and treat sweep time instead as a constraint, which can be set by the user. The question is whether or not this solution can be implemented efficiently.

III. CONSTRAINED LOCAL OPTIMIZATION

With the problems facing MIXER's present optimization feature described, it is now possible to begin work on improvements. The first half of this chapter is dedicated to improving the efficiency of the minefield model calculations and the MOE in Equation 8. A revised objective function with sweep time handled as a constraint, rather than as sweep time cost, is presented. The second half of the chapter details a local optimal search method that solves the constrained version of the revised objective function. This is done to test the accuracy of the above transformations and to provide a possible initial solution for follow-on optimization methods.

A. IMPROVEMENTS TO THE MOE

The optimization problem written in a constrained form (i.e., no Lagrangian multiplier) is

minimize: (10)

$$MOE = \sum_{K=1}^{KMAX} VAL(K) \times LOSS(K) + VAL(KMAX+1) \times SIT$$

subject to: (11)

$$\sum_{J=1}^{JMAX-1} TOTRUN(J) \times HOUR(J) \leq SWEEP\ TIME\ ,$$

where $VAL(KMAX+1)$ is the value of the high value unit (HVU), $LOSS(K)$ is the average number of lost or killed type K resources, and $HOUR(J)$ is the length of time required for one run of sweep type J. Although this form of the problem appears concise, in order to compute SIT and $LOSS(K)$, F must perform

some fairly cumbersome calculations. If any of these can be simplified, then it is worth some investigation.

1. Transformation of Resource Cost

The first transformed term is resource cost, which is

$$resource\ cost = \sum_{K=1}^{KMAX} VAL(K) \times LOSS(K) \ , \tag{12}$$

where $LOSS(K)$ is the number of type K resources destroyed by mines, and $VAL(K)$ is the value of resource type K. The equation for $LOSS(K)$ is

$$LOSS(K) = \sum_{J=1}^{JMAX-1} KILL(J) \times h(K,J) \ , \tag{13}$$

where $KILL(J)$ is the average number of type J sweepers destroyed by mines [Ref. 1: p. 9], and $h(K,J)$ is equal to one if type K resource is killed when type J sweeper is killed and zero otherwise. The equation for $KILL(J)$ is

$$KILL(J) = \sum_{I=1}^{IMAX} MEAN(I) \times TH(I,J) \times Q(I) \times [1 - SURV(I,J)^{TOTRUN(J) \times NP(J)}], \tag{14}$$

where $IMAX$ is the number of different mine types present in the minefield, and $MEAN(I)$ is the user's guess at the total number of mines of type I thought to be in the minefield at the beginning of mine clearance operations. The other terms were defined in the previous chapter.

Instead let

$$S(I,J) = SURV(I,J)^{NP(J)} , \tag{15}$$

where $S(I,J)$ is the probability that a mine of type I remains

with parallel sweeping considered where applicable. Also let

$$MU(I,J) = TH(I,J) \times MEAN(I) \times [\sum_{K=1}^{KMAX} VAL(K) \times h(K,J)] \quad , \tag{16}$$

where $MU(I,J)$ is the average value destroyed by mines of type I for one run by type J minesweeper, assuming all mines are still present. Finally let the clearance level be

$$NQ(I,J) = \prod_{\forall P<J} S(I,P)^{TOTRUN(P)} \quad , \tag{17}$$

where P is all sweep types that have completed their turns in the minefield prior to sweep type J's turn.

With the above definitions of MU and NQ it can be shown that resource cost is

$$resource\ cost = \sum_{I=1}^{IMAX} \sum_{J=1}^{JMAX-1} MU(I,J) \times [NQ(I,J) - NQ(I,J+1)]; \tag{18}$$

a more efficient calculation than Equation 11, and the one to be used from here on.

2. Approximating SIT

The SIT cost term in Equation 10 accounts for simple initial threat to the HVU. Unfortunately the exponential term $KZA(I)/KZB(I)$ in Equation 7 is not an integer and therefore cannot be implemented in a loop. This makes the KATZ approximation for SIT a fatally slow calculation.

It may not be obvious why a speedy calculation for SIT is necessary. A slight digression will offer an explanation. In order to find a better than local optimal solution to the objective function, consider that at some point an exhaustive search may have to be implemented. An exhaustive search will check the total cost for every possible sweep plan within the

limits of the sweep time constraint. An exhaustive search algorithm will be presented in Chapter IV.

If all S sweep types require the same H hours to complete one run, then for the sweep time constraint T

$$N = \frac{[S+(T/H)]!}{S! \times (T/H)!}, \tag{19}$$

where N is number of all feasible solutions. If T is small, an exhaustive search may be completed within a reasonable time frame. As T and/or S increases, N gets big in a hurry, and so do the number of times subroutine F is called and calculations like Equation 7 performed. When H is different for each sweep type, N can increase even faster than in Equation 19.

If SIT can be approximated in a more efficient fashion that maintains an accurate representation of the threat to the HVU, then it would be worth testing. In Equation 7 SIT is approximated as the expected number of lethal mine hits given that the HVU sinks at the first lethal hit. If instead SIT is approximated as the expected number of lethal mine hits given the HVU does not sink then the transformed SIT cost is

$$SIT\ cost = \sum_{I=1}^{IMAX} MU(I, JMAX) \times NQ(I, JMAX), \tag{20}$$

where $JMAX$ is the HVU. Besides avoiding the exponential calculation, this equation follows nicely the format given in Equation 18 for resource cost.

The error suffered for the new approximation of SIT can be determined easily. Figure 6 plots differences between SIT values with the old and new approximations against sweep time constraints. These data points were obtained by executing OPT with the dense scenario in Appendix A.

Figure 6 shows that the difference DELTA decreases rapidly as the sweep time increases, and even with small sweep

Figure 6. *SIT* Difference vs. Sweep Time (hrs.)

times the new approximation is less than .3 away from the KATZ approximation. Assuming that sweeping will continue until SIT is small, the expected number of lethal mine hits provides a good approximation. In addition, the new approximation is consistently larger than the KATZ approximation, making it a more conservative estimation.

3. A Revised Objective Function

The last step in the transformation is to restate the optimization problem with the new MOE. The new problem is

minimize: (21)

$$\sum_{I=1}^{IMAX}\sum_{J=1}^{JMAX-1} MU(I,J)\times[NQ(I,J)-NQ(I,J+1)]+\sum_{I=1}^{IMAX} MU(I,JMAX)\times NQ(I,JMAX)$$

subject to: (22)

$$\sum_{J=1}^{JMAX-1} TOTRUN(J)\times HOUR(J)\leq SWEEP\ TIME\ .$$

B. LOCAL OPTIMAL SEARCH METHOD

In this section a local optimization method for the revised objective function is presented. The difficulty of the problem has increased as feasibility is now an issue. As a result, OPT's movement towards an optimal sweep plan must account for sweep time as well as total cost.

1. Approach

The FORTRAN code for the revised local optimal search method is given in Appendix E. OPT begins by asking the user to input a time constraint. Next OPT must get an initial solution. Since the current plan held in memory may not be feasible, OPT must check the total time it requires. If the current plan is feasible, OPT starts the optimization algorithm with this initial sweep plan. If not OPT simply divides the input time constraint by *JMAX*, the number of sweep types, and gives each sweep type its share of the time. The initial plan is then the number of runs each sweep type can complete in its share of time.

Once the current sweep plan is feasible, OPT begins its search for an optimal plan. OPT proposes new plans by increasing or decreasing runs for each sweep type incrementally, similar to the method described in Figure 1. Decisions regarding choice of a new "best" sweep plan, however, must consider the sweep time. Once OPT determines the proposed sweep plan has an improved total cost, it must insure it is the most time efficient plan. By dividing the cost difference by the sweep time difference resulting from the proposed sweep plan, OPT selects a new sweep plan in a "biggest bang-per-buck" fashion. This method forces OPT to move toward the optimal sweep plan while minimizing movement toward the limit of feasibility.

Once OPT reaches the limit of feasibility it does not

quit. Although all sweep time has been consumed and movements to the feasible limit were done as efficiently as possible, there is a chance that swapping runs between the sweep types will produce a better answer. This can be thought of as moving laterally along the feasibility limit. This movement is also done in a "biggest bang-per-buck" fashion. Once additional lateral moves result in no cost improvement OPT terminates.

2. Solution

To better understand OPT's new local optimization method another sparse scenario is used for illustration. The data files used to generate this scenario are given in Appendix B. The scenario is similar to the one described in Chapter II. Figures 7 and 8 show a surface plot of the MOE without the sweep time cost of Figures 4 and 5.

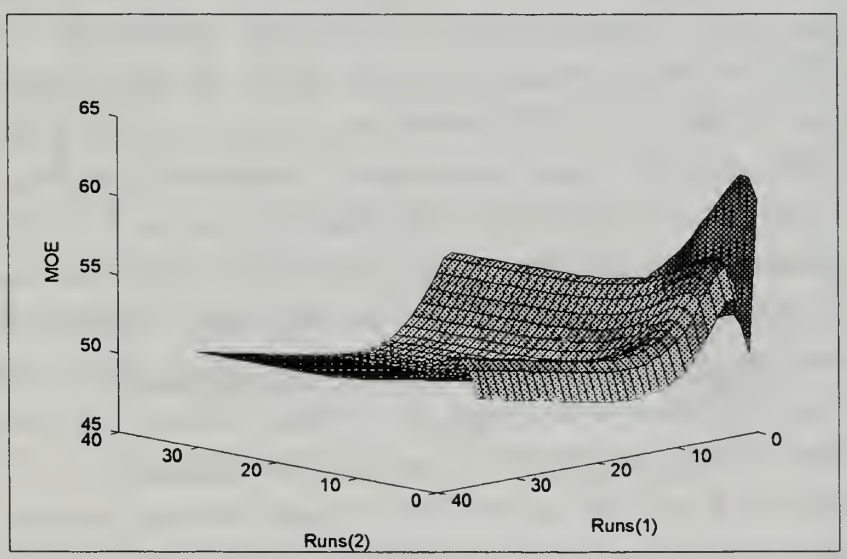

Figure 7. Surface Plot View 1

As can be seen from Figures 7 and 8 a major difference is that the "valley" area is not closed-off but instead continues

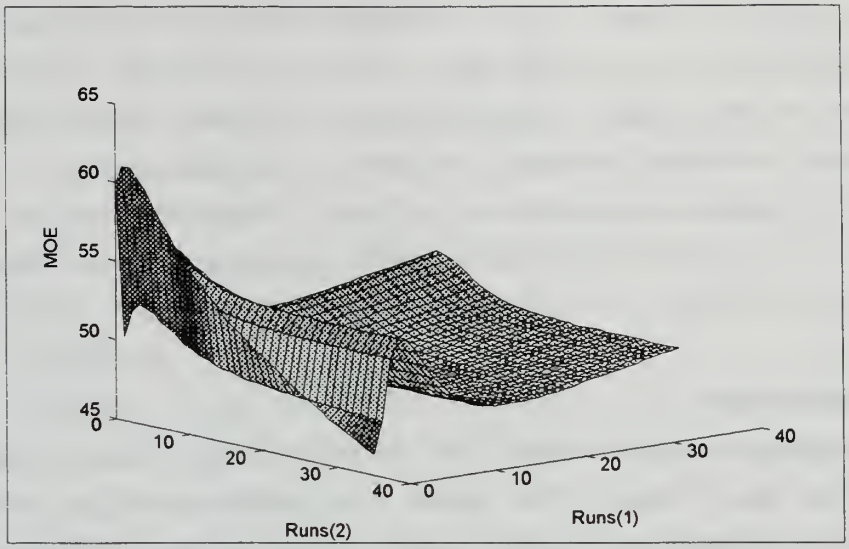

Figure 8. Surface Plot View 2

to slope down away from the Runs (1) axis. Another difference, seen best in Figure 8, is the flapped over edge near the far end of the Runs (2) axis. It is quite obvious from these figures that the constrained objective function is still nonconvex. The time constraint used here is 2600 hours.

Figures 9 and 10 show contour plots of the objective function. The solid lines represent constant values of the MOE. The line of feasibility is plotted as well. This line shows the number of runs for each sweep type beyond which the problem is infeasible due to the sweep time constraint. The feasible region is to the left of this line. Also shown are the local and global minimums for this scenario plotted as circles. The global minimum is 46.288 at the point (35,0).

In Figure 10 actual paths OPT takes during executions of its search algorithm for different initial solutions are presented. Stars indicate randomly choosen initial solutions. As can been seen, infeasible initial solutions are immediately moved to a feasible point.

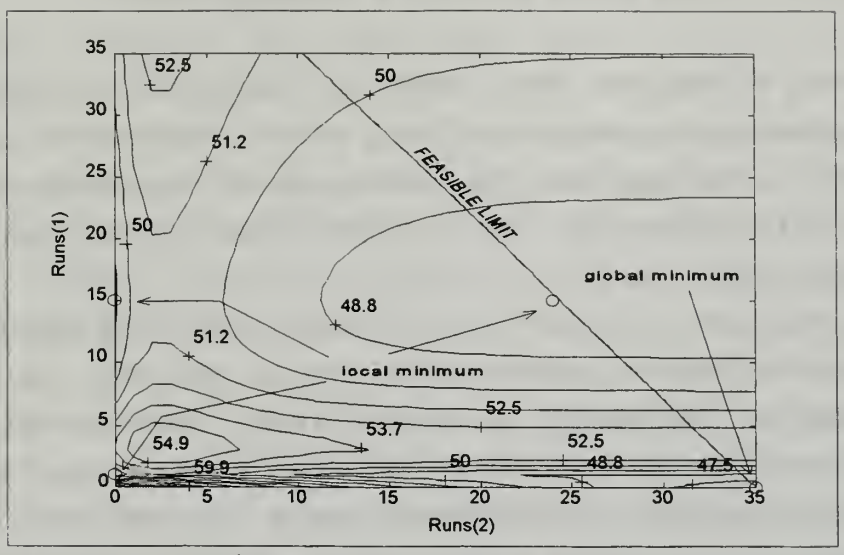

Figure 9. Contour Plot

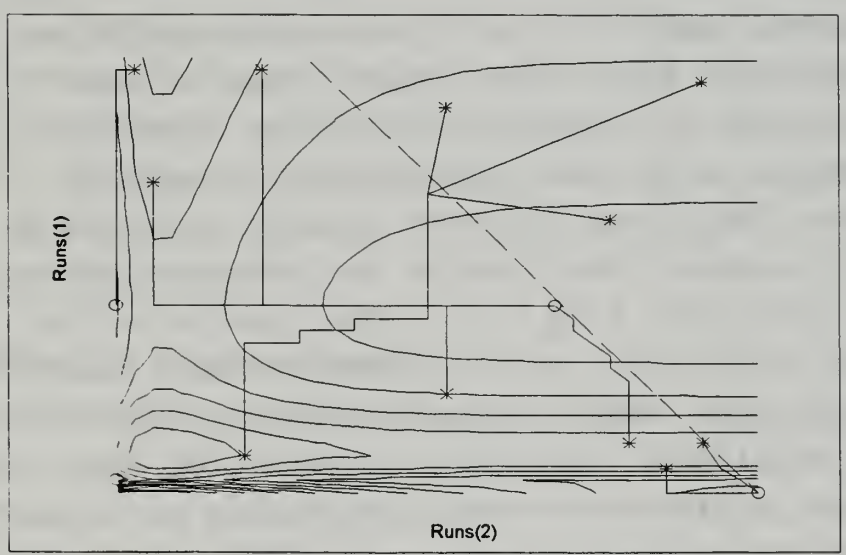

Figure 10. Paths to Local Opt.

Most of the time OPT finds the local minimum in the "valley" area. This is due to the fact that most of the surface area slopes into this region. In the upper lefthand corner, however, the initial solution is on the backside of a ridge that parallels the Runs (1) axis. In this case OPT moves

toward the axis and then down to a local minimum.

In the bottom right hand corner is an initial solution that is very close to the limit of feasibility. Since OPT could not increase sweep runs from here it tries decreasing, which leads to moving down the back side of the ridge running parallel to the Runs (2) axis, and ultimately to the global optimal solution.

Also visible in Figure 10 are cases when OPT reaches the feasible limit before reaching the local optimal. In several of the searches OPT moves laterally along the feasible limit swapping sweep runs at each move. Also notice that some of OPT's movements seem to follow straight lines and then sharply turn towards the minimum point. These tracks display OPT's "biggest bang-per-buck", feature. Unlike in Chapter II where OPT zigzags toward the minimum point, here OPT may stay in one direction until another direction has a more efficient path to a local optimal sweep plan. These figures show that the algorithm works as expected. OPT moves towards the local minimum one run at a time, except when infeasible.

The new OPT allows the user to input the time he has to sweep and produces the locally best sweep plan that can execute in the given time limit. This version of OPT is more flexible by providing a solution for any time constraint the user gives it. The user no longer has to deal with the clumsy Lagrangian multiplier. In addition, tests with this version of OPT on large problems have shown the program is as fast as the Lagrangian version.

IV. SEEKING A BETTER SOLUTION

Although the local optimizer just presented is fast and accurate, the nature of the problem prevents assurance that it finds the global optimal sweep plan. As stated in Chapters I and II the nonconvex problem is extremely difficult to optimize. In fact there is no quick solution that can promise a global optimal answer. In this Chapter two optimization methods will be discussed that may find better then local optimal solutions. The first optimization method is an exhaustive search algorithm. Exhaustive search involves checking the cost of every feasible sweep plan. The second optimization method is a heuristic approach. The method is called simulated annealing because its implementation resembles the annealing process that takes place in the physical world when liquids are cooled to form solids.

The exhaustive search algorithm will be discussed first. A brief description of the algorithm is given, and then a discussion of some of its shortfalls. As may be expected this method, although promising on small problems, is not the ultimate solution method.

A. EXHUASTIVE SEARCH METHOD

The exhaustive search algorithm checks every sweep plan that can occur within the given time constraint. As the algorithm selects plans from the solution set it determines their total costs. Each time a sweep plan is found with an improved total cost that sweep plan is saved as the best sweep plan. Once the testing has exhausted the solution set the algorithm terminates. The improvement this method provides is that it can claim to find the global optimal sweep plan.

The draw back to this method, as mentioned in Chapter III is the actual time the program requires to execute. In Chapter

III the cost calculation was streamlined in the hope that the program's run time could be decreased. Unfortunately the solution set size, *N*, grows as the number of sweep types increases and as the time constraint increases. It is not difficult to imagine a scenario with many sweep types and large sweep time given a large minefield. One example was in the Persian Gulf, where minesweeping efforts that started during the war continued long after the war was over.

1. Approach

The exhaustive search algorithm in OPT checks every possible sweep plan within the sweep time constraint. Once the user has input the sweep time limit, *TLIM*, OPT finds the total cost for no sweeping and sets it equal to *best*. Then OPT begins an algorithm that finds all feasible sweep plans. Each time a feasible sweep plan is found OPT calls the F subroutine and gets its total cost. OPT then compares it with *best*, and if less OPT updates *best*. Then OPT finds the next feasible sweep plan and repeats the process. Once all feasible sweep plans have been compared with *best* OPT terminates, saving the sweep plan that produced *best*. The FORTRAN program for exhaustive search is given in Appendix D.

2. Solution

OPT's exhaustive search finds the global optimal sweep plan for the sparse scenario used in Chapter III, Appendix B. A contour plot of the state space for this scenario is given in Figure 11. This plot shows all the sweep plans OPT checked. OPT required about three seconds to complete an exhaustive search given a time constraint of 2600 hours with this sparse scenario.

In Figure 12 a plot of the actual best plans the exhaustive search algorithm finds along its path to the global

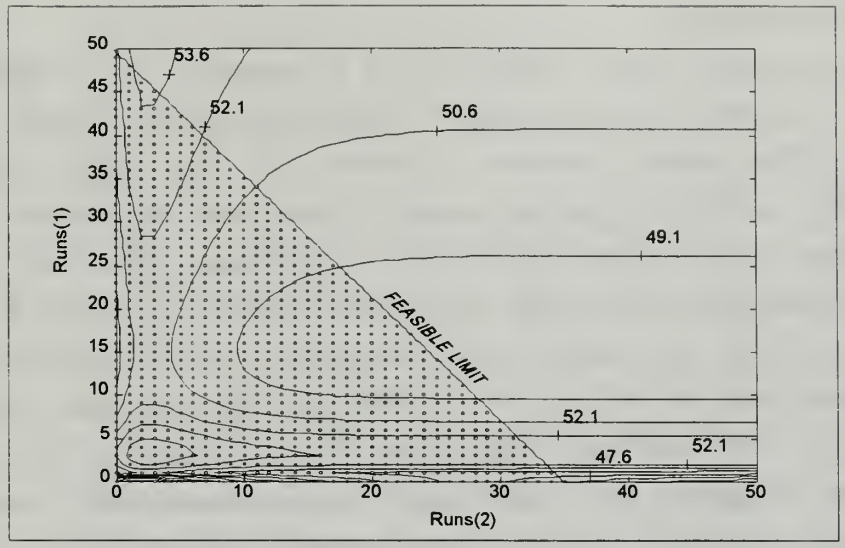

Figure 11. Exhaustive Search

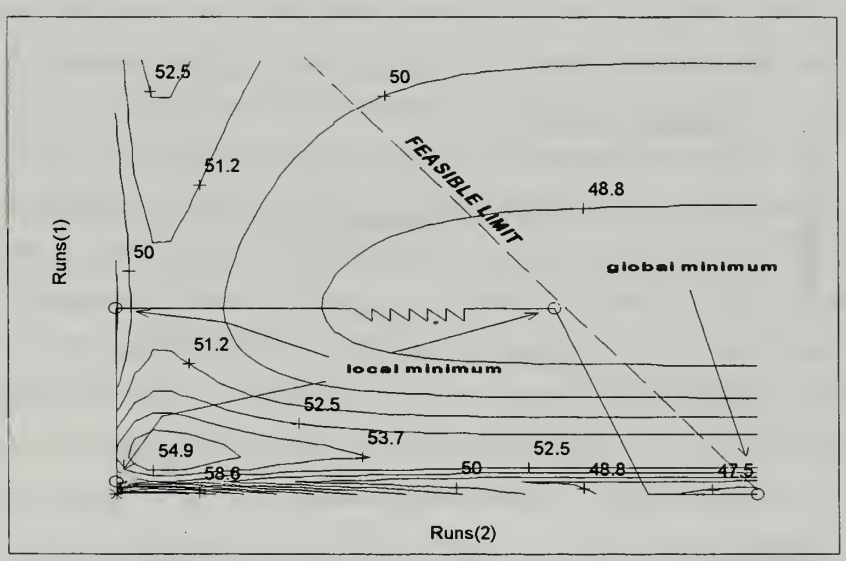

Figure 12. Path to Global Opt.

optimal is given. The borders of the contour plot are suppressed here so that the sections of the path that run along the axes can be seen. Note that the steps taken along the exhaustive search path of best solutions are not always unit length. An example is the step from (0,15) to (20,15).

3. Shortfalls

The two sweep type scenario is an example of a sparse MCM operation. In many cases several different sweep types will be employed. The dense scenario given by the data files in Appendix A, involves seven sweep types and five mine types, which makes it a more difficult problem. The set of all possible feasible solutions for this scenario is much larger then that of the two sweep type scenario. To illistrate Figure 13 compares the solution set sizes of the sparse and dense scenarios for increasing time constraints. Clearly the rate at which the solution set size, *N*, increases for the dense scenario is orders of magnitude greater than for the sparse scenario.

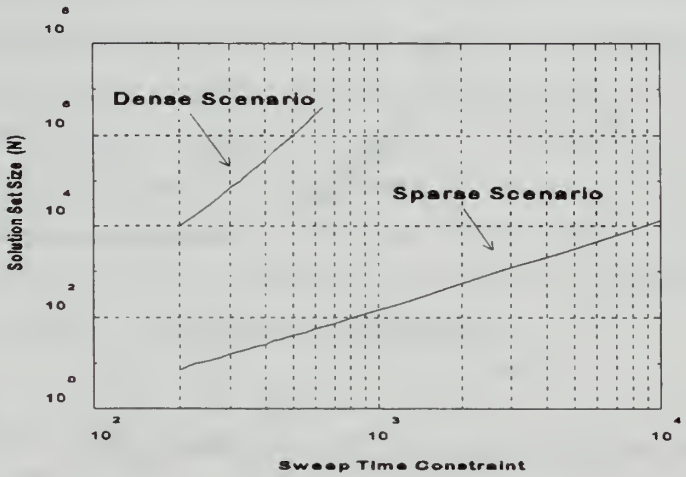

Figure 13. *N* vs. Time Constraint (hrs.)

In Figure 14 exhaustive search run times are plotted against sweep time constraints. Run time data was produced with a UNIX time profiling application [Ref. 6: p. 186].

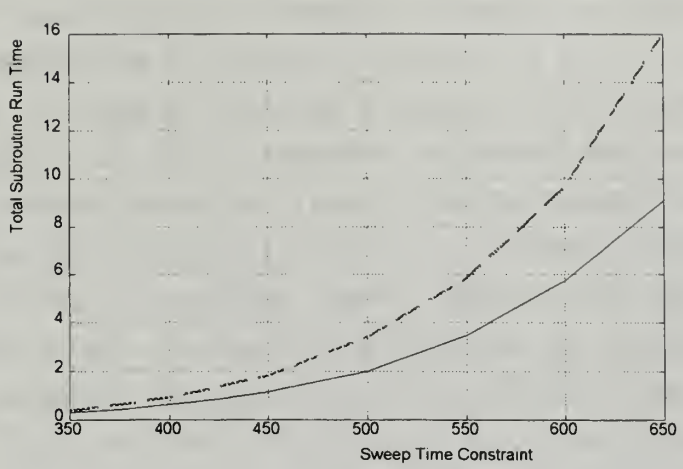

Figure 14. Run Time (min.) vs. Time Constraint (hrs.)

Subroutine F accounts for 93% of total run time during exhaustive search. As can be seen the exhaustive search method quickly becomes too slow for tactical purposes, especially considering that the relatively dense scenario given may actually be a sparse example in real world terms.

Our streamlining efforts in Chapter II were not in vain. The dashed line above represents computer run times for the original computations of subroutine F, and the solid line is for the transformed computations. Figure 14 shows that the new cost computations reduced run times by almost half. Also apparent, however, is that the solution set's growth rate far outmatches the time saved with the improvements. Yet, by pushing the envelope of efficiency there are now a greater number of scenarios that can be exhaustively solved in a reasonable amount of time.

B. SIMULATED ANNEALING METHOD

The local optimal search method in Chapter III finds a solution almost instantaneously, however, it is merely a local minimum. The exhaustive search algorithm finds the global

minimum, but the run time is impractical for many scenarios. The difficulty of the nonconvex problem is such that no search method can guarantee a global optimal solution quickly. A compromise therefore would be a method that has a good chance of finding a better then local optimal solution in a relatively short amount of time. Heuristics encompass a variety of search methods that implement these types of compromises. Simulated annealing is known to be a particularly good approach for the class of optimization problem presented here.

1. What is it?

Simulated annealing is a search method that is based on the cooling of liquids to form solids. In the physical world solids can be formed through annealing, a process by which the material is cooled slowly. If the material is cooled too quickly, energy may be trapped in the solid, forming weak points or irregularities. By cooling at just the right rate most of the trapped energy can escape allowing molecules to stabilize uniformly. This process produces stronger and more pure solids. [Ref. 7]

In the optimizer's world the local optimal search method parallels a process that cools too quickly. The local search always moves downhill if possible. When no downhill moves are present the search terminates. This in a sense freezes the solution in a state that may be less then globally optimal. The simulated annealing search, however, cools more slowly. It allows the search to proceed uphill at times with the prospect of crossing to an adjacent area with a better local minimum. The cooling process, or decrease in temperature, is analogous to a decrease in probability that simulated annealing will allow an uphill move. A simulated annealing run that is cooled slowly will have a greater number of uphill moves. As the

temperature is cooled to a frozen state, the chance for uphill moves approaches zero.

Cooling is conducted by decreasing the temperature in discrete levels. In the physical world a material is held at a constant temperature until sufficient time has passed for all the trapped energy to escape. In the optimization world the probability for uphill moves depends on the temperature level (*TEMP*), which is decreased according to a cooling ratio (*r*). Downhill moves are always accepted when proposed, so once a peak has been crossed the search will not freeze until it has checked the new local minimum. If this local minimum is an improvement over past local minimums then it becomes the best solution and the next hill is attempted. Once the temperature has cooled and the chance for uphill moves decreased to a level that prevents the search from crossing any local peaks, the process is deemed frozen.[Ref.7]

Although simulated annealing can not promise the global optimal solution, its does prevent solutions that are not solely dependent on the local minimum of the initial solution. With each peak that is crossed this search method finds a new locally optimal solution that would have been missed by the local optimization search method. If any of the local minimums found by simulated annealing are better than the one found with the local optimizer, then simulated annealing has accomplished its goal.

2. Solution

As with the local optimal search method, simulated annealing proposes new sweep plans in incremental steps. Once the user has selected simulated annealing, OPT starts the process with an initial solution equal to the local optimal solution found by the local search algorithm. OPT proposes a new solution selected at random from the neighborhood of the

current solution. The neighborhood consists of all solutions that are one sweep run away from the current solution. The proposed solution is then checked for feasibility. Unlike the local search method, simulated annealing can move beyond the feasible limit, but only at an increased cost. If infeasible, a penalty factor (α) is multiplied by the amount of time beyond the constraint the proposed plan requires. This penalty value is then applied to the total cost of the proposed solution.

Next OPT compares the total cost of the proposed solution to that of the current solution. If the proposed solution has a lower cost, a downhill move, OPT accepts the proposed solution by making it the current solution. If the proposed solution has a greater cost, an uphill move, then OPT accepts it with some probability. The penalty factor (α) discussed above must be large enough that an infeasible sweep plan appears to be a large uphill move. The probability of an uphill move depends on the current temperature level and the difference in total cost between the current and proposed *solutions*, *DELTA*, according to the formula

$$PROB\{UPHILL\ MOVE\} = e^{-\Delta/TEMP} \tag{23}$$

[Ref. 7: p. 868].

OPT continues checking solutions in this manner at the current temperature level for a preset length of proposals. This preset temperature length, L, is a function of the average neighborhood size multiplied by a parameter called *SIZEFACTOR*. A full explanation of this and other parameters is given in [Ref. 7]. Once L solutions at the current temperature level have been proposed and either accepted or rejected, OPT decreases the temperature level and begins checking solutions for another temperature. Each time a temperature is completed,

OPT determines the percentage of accepted proposals that occurred. If this percentage is less than the preset parameter *MINPER* then OPT increments a counter *SCNT* by one. If during any temperature level an accepted proposal improves the overall best plan, *CHAMP*, then upon completion of that temperature level, *SCNT* is reset to zero. When *SCNT* equals the preset parameter *FROZ* the process is deemed frozen.

Once the process is frozen OPT checks to see if the last accepted plan was feasible, and if not OPT increases α, decrements *SCNT* by one, and begins a new temperature level. This process continues until the last accepted plan from the last temperature level run is feasible. Then OPT terminates the simulated annealing algorithm saving the best plan found. Once terminated OPT asks the user whether or not further simulated annealing runs are desired. Since the annealing process is a heuristic there is a chance that more improvements can be realized through further iterations. The FORTRAN code for simulated annealing is given in Appendix F.

In Figure 15 a plot the simulated annealing search path for the sparse scenario is presented. This contour plot shows a sample of the accepted solutions during a full annealing run. Many accepted moves are duplicates, since there is nothing preventing simulated annealing from retracing its steps. In this scenario simulated annealing finds the global optimal solution at point (0,35). Examination of Figure 15 shows the explorative nature of the annealing search. Many paths are terminated prior to reaching a boundary. As the search spreads out from the initial solution, marked with a star, it explores the nearest hill and, as is the case here, may successfully cross it. Also shown are moves into the infeasible region, as well as long jumps back to the current best solution which occurs at the beginning of each new temperature level.

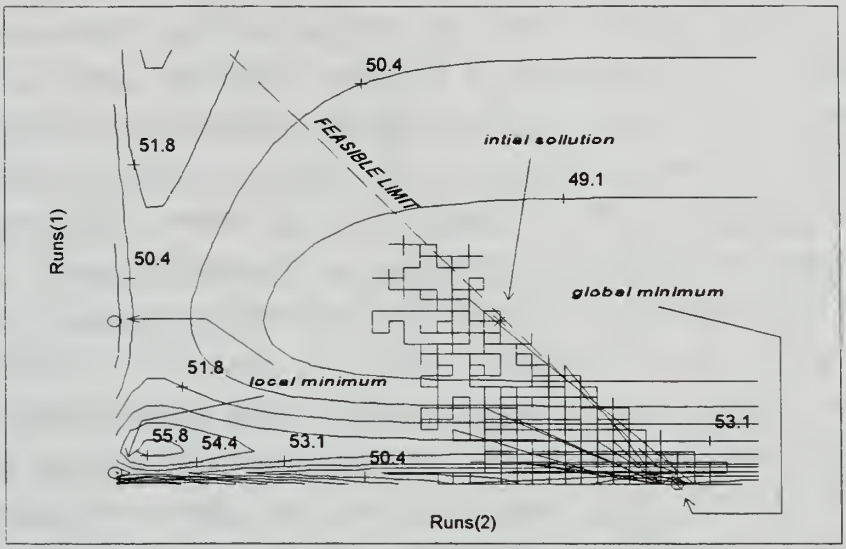

Figure 15. Simulated Annealing

C. COMPARISON TESTS

Two tests compare the local optimal search method to simulated annealing, one with the dense scenario and one with the sparse scenario. Searches were performed by each search method on various versions of each scenario. 120 iterations of both search methods were performed. The scenarios were altered every sixth iteration by randomly generating new actuation probabilities *A(I,J)* (see Equations 1 and 2). At each iteration values for an initial *TOTRUN(J)* between 0 and 10, and sweep time constraint, *TLIM*, between 500 and 1000, were randomly generated.

The data for resource values used in the dense scenario test were slightly modified as shown in Figure 16. *H(K,J)* was also modified for the dense scenario so that damage could occur to sleds used for the *HELCUT* sweep type. These modifications were performed because the author had become too familiar with results for the original values. All other input data remained constant. Each algorithm successfully completed

RELATIVE RESOURCE VALUES VAL(K) FOR OPTIMIZATION, LAST FOR FIRST TRANSIT

2.20 2.00 2.15 0.64 11.50

Figure 16. Modified Resource Values for Dense Scenario

120 searches on each scenario type. Once the local optimizer performed a search it passed its local minimum solution to simulated annealing for possible improvement. The parameters used in the comparison tests are shown in Figure 17. Results of the tests are given in Figure 18.

Parameter	Value
Penalty Factor (α)	0.0005
Temperature Factor (r)	0.95
Size Factor (*SIZEFAC*)	25.0
Minimum acceptance % (*MINPER*)	40.0
Initial Temperature Level (*TEMP*)	$\sum_{K=1}^{KMAX+1} VAL(K)$

Figure 17. Parameter Settings

These tests show that simulated annealing can find improvements to solutions found with the local optimizer. They also reveal that the chance of an improvement occurring and the size of the improvement vary considerably with the scenario. In the dense scenario tests, simulated annealing reduces the total cost 95.8% of the time, with the average reduction being 6.84%. In the sparse scenario tests, simulated annealing reduces the cost 15% of the time, with the average reduction being 35.9%. The dense scenario has a lower average

For 120 Iterations of	Scenario Type Dense	Sparse
Local Optimizer's Mean Cost	2.1599	14.5310
Simulated Annealing's Mean Cost	2.0163	13.0512
# Improvements	115	18
Overall Mean Improvement	0.1463	1.4798
Maximum Improvement	0.9978	31.8443
Overall % Improvement	6.65%	10.18%
% Improvement When Found	6.84%	35.88%

Figure 18. Test Results

cost, but provides considerably more opportunities for improvements. The sparse scenario provides fewer improvements, but the improvements are larger. In one instance of the sparse scenario the local optimizer's best plan cost is 39.3, while simulated annealing finds a solution that costs only 7.5, an improvement of roughly 80%. This level of improvement is on the same order of magnitude as results reported by Johnson and Aragon et al [Ref. 7].

Johnson and Aragon et al discuss efforts to optimize the annealing parameters for the graph partitioning problem. The graph partitioning problem is very different from the objective function found in MIXER. The neighborhood sizes, for example, are much larger in general then those encountered here. Therefore, the optimal *SIZEFAC* setting given in [Ref. 7] is too small for use in OPT. Also the differences in the cost, or cutsize, for any two neighboring solutions in graph partitioning can be orders of magnitude greater then those in OPT's problem. Therefore, the penalty factor for infeasibilty (α) and the initial temperature levels given in [Ref. 7] are not going to be optimal here. Another consideration is that the mine warfare problem is always changing. No two scenarios

are exactly the same. The dynamic environment in which MCM operations are performed may make it nearly impossible to pin down a set of parameters that are optimal in all scenarios. It is suggested that further investigation of parameter settings may show improvements to the results reported above.

The actual run time required for a simulated annealing iteration is a function of the preset parameters. Given a slow enough cooling schedule simulated annealing would, theoretically, check all solutions, as in the exhaustive search. This, however, would take as long if not longer then the exhaustive search, defeating the purpose of the heuristic approach. There exists a trade off between run time and the probability for solution improvement. The parameter settings values should be set to conduct a search that has a reasonably good chance of finding an improvement, and that can be completed in a time frame that is tactically practical. None of the annealing runs performed with the above parameter settings required more then a few seconds to complete.

D. FINAL PRODUCT

An improved version of OPT given in Appendix C attempts to utilizes the best characteristics of each of the search methods developed above; the local optimizer, the exhaustive search, and simulated annealing. When OPT is executed and the time constraint is input, OPT begins by computing the approximate run time that is required to complete an exhaustive search. If this time is less than one minute OPT executes the exhaustive search and displays in table form the total sweep time, SIT, total cost, and average resource losses for the globally optimal sweep plan. If OPT estimates that the exhaustive search will require more then one minute it asks the user to choose between the exhaustive search or local optimization. When local optimization is selected, OPT runs

the local optimizer and then prompts the user to select simulated annealing or to quit. When simulated annealing is selected the annealing run executes. Upon completion the user is asked to choose either further simulated annealing or to quit. After completion of each optimization method OPT outputs the above results.

By combining this set of three search methods, OPT draws from each of their strengths, the speed of the local optimizer, the completeness of the exhaustive search, and the flexibility of simulated annealing, to provide the user with the best sweep plan within the given time constraint. If the user has sufficient opportunity OPT provides the globally best sweep plan for even very large and complicated scenarios. If, however, the user needs an answer quickly OPT can provide the user with the locally best answer instantaneously, and then with just a little more effort OPT can try to improve the locally best plan. The combination of the three methods provides a powerful tool for solving a very difficult problem.

V. CONCLUSION

A. REVIEW

Many of the current mine warfare TDA's fall short of bringing to bear the full potential of today's improved computer speeds and capacities. The TDA MIXER, currently under development, provides a combination of Monte Carlo simulation and analytical optimization techniques with sophistication equal to the challenge. This thesis has shown that by removing sweep time from MIXER's objective function and treating it instead as a constraint, the user has a direct input for sweep time and the local optimal search algorithm developed still provides its optimal solution very quickly.

In addition, this thesis has shown that for MCM operations involving a small number of sweep types and requiring relatively little sweep time, the exhaustive search algorithm, running on a 486 DX2 processor or better, provides a global optimal answer to the objective function in under a minute. Once the objective function's solution size is beyond the capacity for a timely answer by exhaustion, the simulated annealing algorithm developed can be used in conjunction with the local optimizer to provide the best possible sweep plan. Although simulated annealing can not guarantee the global optimal solution, tests have shown that improvement to the local optimal plan can be realized without a large time delay.

B. AREAS FOR FURTHER RESEARCH

Parameter settings for the simulated annealing algorithm require further investigation in order to develop quick methods to optimize their values given a particular MCM scenario. Since the algorithm begins with a locally optimal answer most improvements occur very early in the cooling schedule. A schedule that decreases the temperature level

slowly at first, and then more rapidly as time progresses, may provide some improvement.

The search algorithm for the local optimizer needs further research. At present sweep runs are decreased only in the face of infeasibilty, or when an increase gives a more costly solution. When the current solution rests precisely in the peak of a local optimal area hill, an algorithm that investigates a decrease in sweep runs on par with increases would allow the search to progress down the backside of the hill and possibly to a better local optimal solution. Other areas for improvement are discussed in [Ref. 1: p. 35].

APPENDIX A. DATA FOR DENSE SCENARIO

PARAMS.DAT:

```
MINE TYPES, SWEEP TYPES, RESOURCE TYPES
NOTE THAT THE LAST SWEEP TYPE IS TARGET TRAFFIC
  5  8  4
INDICATOR FOR WHETHER MINE COUNTERS WORK AS COUNTERMEASURE FOR SWEEP
  0  0  1  1  1  1  0  1
WIDTH(YDS), LENGTH(N.MI.), TRAPEZ, HUNT ID TIME(HR.)
    1000.   25.00   .15   .10
NAVIGATION STANDARD ERROR BY SWEEP TYPE
     20.00  20.00  20.00  20.00  30.00  20.00  40.00  60.00
TABLE H(K,J) USAGE OF RESOURCE K PER UNIT OF SWEEP J, 2= VULNERABLE
RESOURCE NAMES READ FROM THIS SECTION
     SHPHNT EODHNT SHPMAG HELMAG SHMGAC HELACU HELCUT
SWEEPER    2     1     2     0     2     0     0
HELICOP    0     0     0     1     0     1     1
EODTEAM    1     2     0     0     0     0     0
   SLED    0     0     0     2     0     2     0
RELATIVE RESOURCE VALUES VAL(K) FOR OPTIMIZATION, LAST FOR FIRST TRANSIT
      2.00  2.00  2.00  0.50  10.00
ORDER OF ENTRY INTO MINEFIELD SEQ(J)
       4     5     6     3     7     2     1
TABLE TURN(J) HOURS PER RUN OF CHANNEL (HOURS)
      .20   .10   .20   .10   .20   .10   .20
TABLE VEL(J) VELOCITY WHILE SWEEPING
      3.0   1.0   5.0  30.0   5.0  30.0  20.0
TABLE DUTY(J) MULTIPLY TIME ON TASK BY DUTY TO GET REAL TIME
      6.0   8.0   4.0  10.0   6.0  10.0  12.0
PROBABILITY ACTUATOR SETTINGS BY MINE TYPE
ACT(I)   0.1   0.5   0.33  0.33  1.0
TABLE A(I,J) SWEEP FRONT (NAMES READ FROM THIS SECTION)
     SHPHNT EODHNT SHPMAG HELMAG SHMGAC HELACU HELCUT TARGET
BOTMAG  200.00 100.00 200.00 100.00 100.00    .00    .00  50.00
BOTACU  100.00 100.00  50.00    .00 100.00 100.00    .00  50.00
BOTPRS  150.00 100.00    .00    .00    .00    .00    .00  20.00
TETMAG  200.00    .00 200.00 200.00 100.00    .00 150.00  70.00
TETCNT  200.00    .00    .00    .00    .00    .00 150.00  70.00
TABLE B(I,J) ACTUATION PROBABILITIES
     SHPHNT EODHNT SHPMAG HELMAG SHMGAC HELACU HELCUT TARGET
BOTMAG   .50   .50   .50   .50   .50   .50  1.00   .50
BOTACU   .50   .50   .50   .50   .50   .50  1.00   .50
BOTPRS   .50   .50   .50   .50   .50   .50  1.00   .50
TETMAG   .50   .50   .50   .50   .50   .50  1.00   .50
TETCNT   .50   .50   .50   .50   .50   .50  1.00   .50
TABLE AF(I,J) DANGEROUS FRONT
     SHPHNT EODHNT SHPMAG HELMAG SHMGAC HELACU HELCUT TARGET
BOTMAG  20.00  10.00  50.00  50.00  10.00    .00    .00  50.00
BOTACU  20.00  20.00  50.00    .00  10.00  50.00    .00  20.00
BOTPRS  20.00  30.00    .00    .00    .00    .00    .00  20.00
TETMAG  30.00    .00  50.00  70.00  10.00    .00    .00  70.00
TETCNT  40.00    .00    .00    .00    .00    .00  10.00  70.00
TABLE BF(I,J) DAMAGE PROBABILITIES CONDITIONAL ON DETONATION
     SHPHNT EODHNT SHPMAG HELMAG SHMGAC HELACU HELCUT TARGET
BOTMAG   .15   .15   .15   .15   .15   .15   .15  1.00
BOTACU   .15   .15   .50   .15   .15   .15   .15  1.00
BOTPRS   .15   .15   .15   .15   .15   .15   .15  1.00
TETMAG   .15   .15   .15   .15   .15   .15   .15  1.00
TETCNT   .15   .15   .15   .15   .15   .15   .15  1.00
```

NUMBERS.DAT:

```
     SHPHNT EODHNT SHPMAG HELMAG SHMGAC HELACU HELCUT TARGET
NTRK(J)     5     3     3     9     5     9     9     9
TRACK POSITIONS FOR EACH OF ABOVE SWEEP TYPES FOLLOW (FORMAT 13I6)
  100   300   500   700   900
  100   500   900
  100   500   900
  100   200   300   400   500   600   700   800   900
  100   300   500   700   900
  100   200   300   400   500   600   700   800   900
  100   200   300   400   500   600   700   800   900
  100   200   300   400   500   600   700   800   900
RUNS PER TRACK PER SWEEPING UNIT FOLLOW: (FORMAT 13I6)
    2     2     2     2     2
    2     2     2
    2     2     2
    2     2     2     2     2     2     2     2     2
    2     2     2     2     2
    2     2     2     2     2     2     2     2     2
    2     2     2     2     2     2     2     2     2
      BOTMAG BOTACU BOTPRS TETMAG TETCNT
AVERAGE  10.00   6.00   6.00   6.00   9.00
STD_DEV  20.00   6.00   3.00   1.00   3.00
     SWEEPER HELICOP EODTEAM  SLED
RESOURC     2      2      2      2
```

APPENDIX B. DATA FOR SPARSE SCENARIO

PARAMS.DAT:

```
MINE TYPES, SWEEP TYPES, RESOURCE TYPES
NOTE THAT THE LAST SWEEP TYPE IS TARGET TRAFFIC
  6  3  4
INDICATOR FOR WHETHER MINE COUNTERS WORK AS COUNTERMEASURE FOR SWEEP
  0  1
WIDTH(YDS), LENGTH(N.MI.), TRAPEZ, HUNT ID TIME(HR.)
    1000.   45.00   .01  0.00
NAVIGATION STANDARD ERROR BY SWEEP TYPE
     20.00  40.00  60.00
TABLE H(K,J) USAGE OF RESOURCE K PER UNIT OF SWEEP J, 2= VULNERABLE
RESOURCE NAMES READ FROM THIS SECTION
     HELACU HELCUT
SWEEPER    0     0
HELICOP    1     1
EODTEAM    0     0
   SLED    2     2
RELATIVE RESOURCE VALUES VAL(K) FOR OPTIMIZATION, LAST FOR FIRST TRANSIT
     10.00  10.00  10.00  10.00  10.00
ORDER OF ENTRY INTO MINEFIELD SEQ(J)
       1    2
TABLE TURN(J) HOURS PER RUN OF CHANNEL (HOURS)
      .15   .35
TABLE VEL(J) VELOCITY WHILE SWEEPING
     19.0  13.6
TABLE DUTY(J) MULTIPLY TIME ON TASK BY DUTY TO GET REAL TIME
     20.8  20.3
PROBABILITY ACTUATOR SETTINGS BY MINE TYPE
ACT(I)   1.0   1.0   1.0   0.33  1.0   1.0
TABLE A(I,J) SWEEP FRONT (NAMES READ FROM THIS SECTION)
     HELACU HELCUT TARGET
BOTMAG  744.00 150.00 650.00
BOTACU   55.00 895.00  50.00
BOTPRS  400.00 999.00 643.00
TETMAG  420.00 180.00  70.00
TETCNT   87.00 305.00 240.00
FLTCNT   25.00 690.00 100.00
TABLE B(I,J) ACTUATION PROBABILITIES
     HELACU HELCUT TARGET
BOTMAG   .92   .07   .95
BOTACU   .15   .00   .00
BOTPRS   .30   .00   .83
TETMAG   .69   .80   .00
TETCNT   .00   .35   .34
FLTCNT   .00  0.38  1.00
TABLE AF(I,J) DANGEROUS FRONT
        HELACU HELCUT TARGET
BOTMAG     .00    .00 300.00
BOTACU   38.00 139.00  20.00
BOTPRS     .00    .00 290.00
TETMAG  617.00   8.00  70.00
TETCNT   35.00  87.00 145.00
FLTCNT    5.00 130.00  10.00
```

```
TABLE BF(I,J) DAMAGE PROBABILITIES CONDITIONAL ON DETONATION
     HELACU HELCUT TARGET
BOTMAG   .00    .00    .88
BOTACU   .10    .00    .00
BOTPRS   .00    .00    .70
TETMAG   .63    .00    .00
TETCNT   .00    .00    .83
FLTCNT   .00    .65    .00
```

NUMBERS.DAT:

```
     HELACU HELCUT TARGET
NTRK(J)     9     9     9
TRACK POSITIONS FOR EACH OF ABOVE SWEEP TYPES FOLLOW (FORMAT 13I6)
  100   200   300   400   500   600   700   800   900
  100   200   300   400   500   600   700   800   900
  100   200   300   400   500   600   700   800   900
RUNS PER TRACK PER SWEEPING UNIT FOLLOW: (FORMAT 13I6)
   2     2     2     2     2     2     2     2     2
   2     2     2     2     2     2     2     2     2
     BOTMAG BOTACU BOTPRS TETMAG TETCNT FLTCNT
AVERAGE  10.00   6.00   6.00   6.00   9.00    6.00
STD_DEV  20.00   6.00   3.00   1.00   3.00   12.00
    SWEEPER HELICOP EODTEAM  SLED
RESOURC     2     2     2     2
```

APPENDIX C. SUBROUTINE OPT

```
      SUBROUTINE OPT(
     I         KZA,KZB,TH,SURV,NP,MEAN
     O         ,SIT,LOSS,TIME,HUNT
     U         ,TOTRUN,SEED)
      IMPLICIT NONE
C LIMITS ON MINE TYPES, SWEEP TYPES, MINES, RESOURCES, TRACKS
      INTEGER*4 O1,O2,O4,O5
      PARAMETER(O1=10,O2=9,O4=8,O5=40)
      REAL*4 KZA(O1),KZB(O1),A(O1,O2),B(O1,O2),AF(O1,O2),WIDTH,TR,HUNT
     +    ,BF(O1,O2),HOUR(O2),SURV(O1,O2),SIG(O2),COST,SIT,TIME,HT
     +    ,ACT(O1),TH(O1,O2),LOSS(O4),MEAN(O1),Q(O2),VAL(O4)
     +    ,SUMVAL(O2),S(O1,O2),MU(O1,O2),TLIM,MINH,AVGH,TCONV,EST
      CHARACTER NM(O1)*7,NS(O2)*8,NR(O4)*7
      INTEGER*4 IMAX,JMAX,KMAX,JM1,NP(O2),TMP
     + ,IND(O2),SEQ(O2),H(O4,O2),I,J,K,TOTRUN(O2),FLAG
     + ,MAXINT,OPTYPE,SEED
      COMMON A,B,AF,BF,HOUR,ACT,WIDTH,TR,SIG,VAL,HT
     +     ,SEQ,IMAX,JMAX,KMAX,IND,H
     +     ,NM,NS,NR
      DATA MAXINT  /2147483647/

      JM1=JMAX-1

C PRE-COMPUTE S(I,J) AND MU(I,J)
      DO 60 J=1,JM1
        SUMVAL(J)=0.
        DO 40 K=1,KMAX
         IF(H(K,J).EQ.2)THEN
          SUMVAL(J)=SUMVAL(J)+VAL(K)
         ENDIF
40    CONTINUE
       DO 50 I=1,IMAX
        S(I,J)=SURV(I,J)**NP(J)
        MU(I,J)=TH(I,J)*MEAN(I)*SUMVAL(J)
50    CONTINUE
60    CONTINUE
      DO 70 I=1,IMAX
       MU(I,JMAX)=TH(I,JMAX)*MEAN(I)*VAL(KMAX+1)
70    CONTINUE

C ++++++++++REMOVE COMMENT TO WRITE S AND MU TO SCREEN++++
C     DO 75 I=1,IMAX
C       WRITE(6,80)(S(I,J),J=1,JM1)
C 75  CONTINUE
C     WRITE(6,80)
C 80  FORMAT(10F8.4)
C     DO 85 I=1,IMAX
C       WRITE(6,90)(MU(I,J),J=1,JMAX)
C 85  CONTINUE
C     WRITE(6,90)
C 90  FORMAT(10F8.4)
C ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

C INPUT SWEEP TIME CONSTRAINT
100   WRITE(6,*)'INPUT TIME, OR 0 FOR MAIN MENU: '
```

```
      READ(5,*)TLIM
      WRITE(6,*)
      IF(TLIM.LE.0)RETURN
      MINH=MAXINT
      DO 110 J=1,JM1
        IF(HOUR(J).LT.MINH)THEN
         MINH=HOUR(J)
        ENDIF
110   CONTINUE
      IF(TLIM.LT.MINH)THEN
       WRITE(6,120)MINH
120    FORMAT('TIME TOO SMALL, NEED AT LEAST ',F6.2,' HOURS!')
       GO TO 100
      ENDIF

C ESTIMATE TIME FOR EXHAUSTIVE SEARCH BASED ON POLYNOMIAL
C APPROXIMATION FOR SCENARIO WITH 7 SWEEP TYPES AND A MEAN
C HOUR(J) OF 48.6

C CONVERT CASE SCENARIO AND FLAG IF < 1 MIN
      OPTYPE=0
      AVGH=0
      DO 130 J=1,JM1
        AVGH=AVGH+HOUR(J)
130   CONTINUE
      TCONV=(48.6/AVGH)*TLIM*JM1
      EST=(.0003*TCONV*TCONV)-(.22*TCONV)+40.
      IF(JM1.LE.7)THEN
       IF(TCONV.LT.375.0.OR.JM1.LT.3)THEN
        OPTYPE=1
       ELSE
        EST=EST**(JM1/7.)
       ENDIF
      ELSE
       EST=(EST*(48.6/AVGH)*JM1)**(JM1/7.)
      ENDIF
      IF(OPTYPE.EQ.0)THEN
       IF(EST.LE.1.)THEN
        OPTYPE=1
       ENDIF
      ENDIF
      IF(OPTYPE.EQ.1)THEN
       WRITE(6,*)'EXHAUSTIVE SEARCH IN < 1 MINUTE'
      ENDIF

C ++++++REMOVE COMMENT TO SKIP AUTO EXHAUSTIVE SEARCH++++++
C   OPTYPE=0
C ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

C SELECT OPTIMIZATION METHOD
      IF(OPTYPE.EQ.0)THEN
       WRITE(6,140)EST
140     FORMAT('EXHAUSTIVE SEARCH IN ABOUT ',F7.2,' MINUTES')
       WRITE(6,*)' CHOOSE: OPTIMIZE(0), EXHAUSTIVE SEARCH(1)'
       READ(5,*)OPTYPE
      ENDIF

      IF(OPTYPE.EQ.1)THEN
       WRITE(6,*)'EXHAUSTIVE SEARCH'
```

```
     CALL EXH(S,MU,TLIM,TOTRUN)
    ELSE
     WRITE(6,*)'LOCAL OPTIMUM'
     CALL LOPT(S,MU,TLIM,TOTRUN)

    ENDIF
    FLAG=0

C ++++++++REMOVE COMMENT FOR KATZ SIT++++++++++++++++++++++++++
C FLAG=1
C ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

150   CALL F(TOTRUN,NP,TH,SURV,KZA,KZB,MEAN,JM1,FLAG
    +,LOSS,COST,SIT,Q,HUNT)
    TIME=0
    DO 160 J=1,JM1
    TIME=TIME+TOTRUN(J)*HOUR(J)
160   CONTINUE
    TIME=TIME+HUNT*HT
    WRITE(6,170)TIME,SIT,COST
170   FORMAT(' THEORETICAL RESULTS ARE TIME(HRS):',F8.2,' SIT:',F6.4,
    +' COST:',F8.4)
    WRITE(6,*)'AVERAGE LOSSES NEXT:'
    WRITE(6,180)(NR(K),LOSS(K),K=1,KMAX)
180   FORMAT(5(A7,':',F6.2,2X))

    TMP=0
    IF(OPTYPE.EQ.0)THEN
     WRITE(6,*)'FURTHER SIMULATED ANNEALING MAY IMPROVE COST'
     WRITE(6,*)'CHOOSE: QUIT(0), SIMULTED ANNEALING(1)'
     READ(5,*)TMP
     IF(TMP.EQ.1)THEN
      WRITE(6,*)'SIMULATED ANNEALING'
      CALL SIM(S,MU,TLIM,TOTRUN,SEED)

      GO TO 150
     ENDIF
    ENDIF
    GO TO 100
    END

    SUBROUTINE F(
    I        TOTRUN,NP,TH,SURV,KZA,KZB,MEAN,JM1,FLAG
    O        ,LOSS,COST,SIT,Q,HUNT)
    IMPLICIT NONE
    INTEGER*4 O1,O2,O4
    PARAMETER(O1=10,O2=9,O4=8)
    REAL*4 KZA(O1),KZB(O1),A(O1,O2),B(O1,O2),AF(O1,O2),WIDTH,TR,HUNT
    +    ,BF(O1,O2),HOUR(O2),SURV(O1,O2),SIG(O2),FAC,TMP,VAL(O4)
    +    ,ACT(O1),TH(O1,O2),HT
    REAL*4 LOSS(O4),COST,SIT,KILL(O2),Q(O2),MEAN(O1)
    CHARACTER NM(O1)*7,NS(O2)*8,NR(O4)*7
    INTEGER*4 IMAX,JMAX,KMAX,J,JM1,TOTRUN(O2),FLAG
    + ,IND(O2),SEQ(O2),H(O4,O2),I,K,JP,R,NP(O2)
    COMMON A,B,AF,BF,HOUR,ACT,WIDTH,TR,SIG,VAL,HT
    +     ,SEQ,IMAX,JMAX,KMAX,IND,H
    +     ,NM,NS,NR
    COST=0.
    HUNT=0.
```

```
      DO 5 K=1,KMAX
        LOSS(K)=0.
5     CONTINUE
      DO 8 J=1,JM1
8     KILL(J)=0
      DO 10 I=1,IMAX
10    Q(I)=1.
      DO 40 JP=1,JM1
      J=SEQ(JP)
C CALCULATE LETHAL HITS ON TYPE J SWEEPS
      R=TOTRUN(J)*NP(J)
      DO 20 I=1,IMAX
        FAC=SURV(I,J)**R
        TMP=MEAN(I)*Q(I)*(1.-FAC)
        IF(IND(J).EQ.0)HUNT=HUNT+TMP
        KILL(J)=KILL(J)+TH(I,J)*TMP
        Q(I)=Q(I)*FAC
20    CONTINUE
C TRANSLATE SWEEP KILLS TO RESOURCE KILLS
      DO 30 K=1,KMAX
      IF(H(K,J).EQ.2)THEN
        LOSS(K)=LOSS(K)+KILL(J)
        COST=COST+VAL(K)*KILL(J)
      ENDIF
30    CONTINUE
40    CONTINUE
C CALCULATE SIT
      SIT=0.
      DO 50 I=1,IMAX
        TMP=TH(I,JMAX)*Q(I)
        SIT=SIT+MEAN(I)*TMP
50    CONTINUE

      IF(FLAG.EQ.1)THEN
C CALCULATE SIT USING KATZ GENERATING FUNCTION
      SIT=1.
      DO 60 I=1,IMAX
        TMP=TH(I,JMAX)*Q(I)
        FAC=KZB(I)
        IF(ABS(FAC).GT..001)THEN
         SIT=SIT*(1+FAC*TMP/(1.-FAC))**(-KZA(I)/FAC)
        ELSE
         SIT=SIT*EXP(-KZA(I)*TMP)
        ENDIF
60    CONTINUE
      SIT=1.-SIT
      ENDIF
      COST=COST+VAL(KMAX+1)*SIT
      END
```

APPENDIX D. EXHAUSTIVE SEARCH SUBROUTINE

```
      SUBROUTINE EXH(
     I         S,MU,TLIM
     U         ,TOTRUN)
      IMPLICIT NONE
C LIMITS ON MINE TYPES, SWEEP TYPES, MINES, RESOURCES, TRACKS
      INTEGER*4 O1,O2,O4,O5
      PARAMETER(O1=10,O2=9,O4=8,O5=40)
      REAL*4 A(O1,O2),B(O1,O2),AF(O1,O2),WIDTH,TR
     + ,BF(O1,O2),HOUR(O2),SIG(O2),COST,TIME,HT
     + ,S(O1,O2),BEST,ACT(O1),TLIM,MU(O1,O2),VAL(O4)
      CHARACTER NM(O1)*7,NS(O2)*8,NR(O4)*7
      INTEGER*4 IMAX,JMAX,KMAX,JM1,MAXRND,X(O2)
     + ,IND(O2),SEQ(O2),H(O4,O2),J,JP,TOTRUN(O2)
      COMMON A,B,AF,BF,HOUR,ACT,WIDTH,TR,SIG,VAL,HT
     +     ,SEQ,IMAX,JMAX,KMAX,IND,H
     +     ,NM,NS,NR

      JM1=JMAX-1
C MAXROUND GIVES THE USER AN ESTIMATION OF PROGRESS
      MAXRND=INT(TLIM/HOUR(JM1))+1
C INTIALIZE RUNS TO 0 FOR ALL SWEEP TYPES
      DO 10 J=1,JM1
        TOTRUN(J)=0
10      X(J)=0
C INITIALIZE BEST, THE MINIMUM COST
      CALL NF1(X,S,MU,BEST)
      J=1
      TIME=0
40    X(J)=X(J)+1
      TIME=TIME+HOUR(J)
      IF(TIME.GT.TLIM)THEN
        X(J)=0
        J=J+1
        IF(J.GT.JM1)THEN
         GO TO 50
        ELSEIF(J.EQ.JM1)THEN

C ++++++REMOVE COMMENT TO WATCH ROUNDS COMPLETE++++++++++++++
C       WRITE(6,*)'FINISHED ROUND ',X(J)+1,' OF ',MAXRND
C ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

        ENDIF
C PREVENT ROUNDOFF ACCUMULATION IN TIME BY AVOIDING SUBTRACTION
        TIME=0
        DO 20 JP=J,JM1
20       TIME=TIME+X(JP)*HOUR(JP)
        GO TO 40
      ELSE
        CALL NF1(X,S,MU,COST)
        IF(BEST.GT.COST)THEN
         BEST=COST
         DO 30 JP=1,JM1
30        TOTRUN(JP)=X(JP)
        ENDIF
        J=1
```

```
      GO TO 40
    ENDIF
50  CONTINUE
    END

    SUBROUTINE NF1(
    I         TOTRUN,S,MU
    O         ,COST)
    IMPLICIT NONE
    INTEGER*4 O1,O2,O4
    PARAMETER(O1=10,O2=9,O4=8)
    REAL*4 A(O1,O2),B(O1,O2),AF(O1,O2),WIDTH,TR,SP
    + ,BF(O1,O2),HOUR(O2),SIG(O2),MU(O1,O2)
    + ,ACT(O1),HT,COST,Q(O1),S(O1,O2),VAL(O4)
    CHARACTER NM(O1)*7,NS(O2)*8,NR(O4)*7
    INTEGER*4 IMAX,JMAX,KMAX,TOTRUN(O2)
    + ,IND(O2),SEQ(O2),H(O2,O4),I,JP,LJ
    COMMON A,B,AF,BF,HOUR,ACT,WIDTH,TR,SIG,VAL,HT
    +     ,SEQ,IMAX,JMAX,KMAX,IND,H
    +     ,NM,NS,NR

    COST=0.
    DO 150 I=1,IMAX
       Q(I)=1.
150  CONTINUE
    DO 155 JP=2,JMAX
      LJ=SEQ(JP-1)
      DO 160 I=1,IMAX
       SP=S(I,LJ)**TOTRUN(LJ)
       COST=COST+(MU(I,LJ)*(Q(I)*(1-SP)))
       Q(I)=Q(I)*SP
160   CONTINUE
155   CONTINUE
    DO 170 I=1,IMAX
      COST=COST+(MU(I,JMAX)*Q(I))
170   CONTINUE
    END
```

APPENDIX E. LOCAL OPTIMIZATION SUBROUTINE

```
      SUBROUTINE LOPT(
     I         S,MU,TLIM
     U         ,TOTRUN)
      IMPLICIT NONE
C LIMITS ON MINE TYPES, SWEEP TYPES, MINES, RESOURCES, TRACKS
      INTEGER*4 O1,O2,O4,O5
      PARAMETER(O1=10,O2=9,O4=8,O5=40)
      REAL*4 A(O1,O2),B(O1,O2),AF(O1,O2),WIDTH,TR
     +    ,BF(O1,O2),HOUR(O2),SIG(O2),COST,HT
     +    ,ACT(O1),TMPC,VAL(O4),S(O1,O2),MU(O1,O2)
     +    ,TOTIME,TLIM,PERTIME,PERBANG,SPECMUL
     +    ,BESTBANG,OVERTIME,PERBANG2,BESTBANG2,TIMEFAC
      CHARACTER NM(O1)*7,NS(O2)*8,NR(O4)*7
      INTEGER*4 IMAX,JMAX,KMAX,JM1,FEAS,BESTJ
     + ,IND(O2),H(O4,O2),J,TOTRUN(O2),BESTRUN(O2)
     + ,TRIAL,BESTJD,BESTJDD,BESTJDD1,HOLDD,MINUSRUN
     + ,HOLDDD,HOLDDD1,HOLD,JP,FLAG,FLAG3,FLAG2,SEQ(O2)
      COMMON A,B,AF,BF,HOUR,ACT,WIDTH,TR,SIG,VAL,HT
     +     ,SEQ,IMAX,JMAX,KMAX,IND,H
     +     ,NM,NS,NR

      JM1=JMAX-1
      FEAS=0
C CALCULATE INITIAL TOTAL TIME
      TOTIME=0.
      DO 10 J=1,JM1
      TOTIME=TOTIME+TOTRUN(J)*HOUR(J)
10    CONTINUE

C CHECK TO SEE IF INITIAL PLAN IS FEASIBLE
C AND IF NOT GET A FEASIBLE PLAN
20    IF(TOTIME.LE.TLIM)THEN
      ELSE
       PERTIME=TLIM/(REAL(JM1))
       TOTIME=0.
       DO 30 J=1,JM1
         TOTRUN(J)=INT(PERTIME/HOUR(J))
         TOTIME=TOTIME+HOUR(J)*TOTRUN(J)
30    CONTINUE
      ENDIF
      TOTRUN(JMAX)=1
C INITIALIZE TMPC, THE MINIMUM COST
      CALL NF1(TOTRUN,S,MU,TMPC)

C +++++++REMOVE COMMENT TO WRITE INTIAL PLAN TO SCREEN++++++++++
C     WRITE(6,*)'INITIAL PLAN'
C     WRITE(6,*)'TIME =',TOTIME
C     WRITE(6,*)(TOTRUN(J),J=1,JMAX)
C     WRITE(6,*)(HOUR(J),J=1,JM1)
C     WRITE(6,*)'COST = ',TMPC
C ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

C SET INITIAL BEST RUNS
      DO 40 J=1,JMAX
        BESTRUN(J)=TOTRUN(J)
```

```
40  CONTINUE
    PERBANG=0.
    PERBANG2=0.
    TRIAL=0
    SPECMUL=0.

C BEGIN LOCAL SEARCH
50  FLAG=0
    FLAG2=0
    FLAG3=0
    TRIAL=TRIAL+1
    BESTBANG=0.
    BESTBANG2=-1.
C SET TRIAL BEST RUNS
    TOTIME=0
    DO 60 J=1,JM1
      TOTRUN(J)=BESTRUN(J)
      TOTIME=TOTIME+TOTRUN(J)*HOUR(J)
60  CONTINUE
C SET TRIAL BEST COST
    CALL NF1(TOTRUN,S,MU,TMPC)

C SET SPECIAL MULTIPLIER FOR CASES OF IMPROVED
C COST THROUGH DECREASE IN SWEEP EFFORT
    IF(TOTIME.GT.0)THEN
     SPECMUL=TMPC/TOTIME
    ELSE
     SPECMUL=0.
    ENDIF

C +++++++REMOVE COMMENT TO WRITE TRIAL RESULTS TO SCREEN+++++
C    WRITE(6,*)'TRIAL # ',TRIAL
C    WRITE(6,*)'TIME =',TOTIME
C    WRITE(6,*) (TOTRUN(J),J=1,JMAX)
C    WRITE(6,*) 'CURRENT BEST COST = ',TMPC
C++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

    DO 90 J=1,JM1

C RESET RUNS AND TIME TO LAST BEST
    TOTIME=0
    DO 70 JP=1,JM1
       TOTRUN(JP)=BESTRUN(JP)
       TOTIME=TOTIME+HOUR(JP)*TOTRUN(JP)
70  CONTINUE

C INCREASE SWEEP RUN J BY 1
    TOTRUN(J)=TOTRUN(J)+1
    TOTIME=TOTIME+HOUR(J)

C CHECK IF FEASIBLE
    IF(TOTIME.LE.TLIM)THEN
     CALL NF1(TOTRUN,S,MU,COST)

C INCREASE IN TIME W/O A SWAP
C CHECK IF BETTER THAN BEST BANG FOR
C THE BUCK
      IF(COST.LT.TMPC)THEN
       PERBANG=(TMPC-COST)/HOUR(J)
```

```
      IF(PERBANG.GT.BESTBANG)THEN
       BESTBANG=PERBANG
       BESTJ=J
       HOLD=TOTRUN(J)
       FLAG=1
       FLAG2=0
      ENDIF

C WHEN NOT BETTER TRY DECREASING SWEEP RUN
     ELSEIF(TOTRUN(J).GT.1)THEN
        TOTIME=TOTIME-2*HOUR(J)
        TOTRUN(J)=TOTRUN(J)-2
        CALL NF1(TOTRUN,S,MU,COST)

C DECREASE IN TOTIME W/O A SWAP
C AND CHECK IF BETTER
        IF(COST.LT.TMPC)THEN
         PERBANG2=TMPC-COST+(HOUR(J)*SPECMUL)
         IF(PERBANG2.GT.BESTBANG2)THEN
          BESTBANG2=PERBANG2
          FLAG3=1
          BESTJDD=J
          HOLDDD=TOTRUN(J)

C SINCE NO SWAP LET SECOND CHANGE REPEAT
C FIRST CHANGE
          HOLDDD1=TOTRUN(J)
          BESTJDD1=J
         ENDIF
        ENDIF
     ELSE
     GO TO 90
     ENDIF

C WHEN NOT FEASIBLE
   ELSE

C COMPUTE AMOUNT OF INFEASIBILITY
     OVERTIME=TOTIME-TLIM

C TRY REDUCE OTHER SWEEP RUNS AND
C CHECK IF A BETTER PLAN
     DO 80 JP=1,JM1

C FOR OTHER SWEEPS COMPUTE RUNS NEED TO
C REDUCE TO MAKE FEASIBLE
      IF(JP.NE.J)THEN
       MINUSRUN=1+INT(OVERTIME/HOUR(JP))

C WHEN ABLE TO REDUCE REQUIRED RUNS CHECK COST
C OF PLAN
       IF(TOTRUN(JP).GE.MINUSRUN)THEN
        TOTRUN(JP)=TOTRUN(JP)-MINUSRUN
        CALL NF1(TOTRUN,S,MU,COST)

C W/SWAP
C CHECK IF BETTER
        IF(COST.LT.TMPC)THEN
         TIMEFAC=HOUR(J)-MINUSRUN*HOUR(JP)
```

```
C DECREASE IN TIME W/SWAP
C WHEN BETTER AND TOTIME DECREASES (OR UNCHANGED) SET UP
C BEST OF BEST CHECK
         IF(TIMEFAC.LE.0)THEN
          PERBANG2=TMPC-COST-(TIMEFAC*SPECMUL)
          IF(PERBANG2.GT.BESTBANG2)THEN
           FLAG3=1
           BESTBANG2=PERBANG2
           BESTJDD1=JP
           BESTJDD=J
           HOLDDD1=TOTRUN(JP)
           HOLDDD=TOTRUN(J)
          ENDIF

C INCREASE IN TIME W/SWAP
C WHEN TOTIME INCREASES DO A STANDARD CHECK
C OF BEST
         ELSE
          PERBANG=(TMPC-COST)/TIMEFAC
          IF(PERBANG.GT.BESTBANG)THEN
           FLAG2=1
           FLAG=0
           BESTBANG=PERBANG
           BESTJ=J
           BESTJD=JP
           HOLD=TOTRUN(J)
           HOLDD=TOTRUN(JP)
          ENDIF
         ENDIF
        ENDIF
       ENDIF
      ENDIF
80  CONTINUE
   ENDIF
90  CONTINUE
   IF(FLAG3.EQ.1)THEN
    BESTRUN(BESTJDD)=HOLDDD
    BESTRUN(BESTJDD1)=HOLDDD1
    GO TO 50
   ELSEIF(FLAG2.EQ.1)THEN
    BESTRUN(BESTJ)=HOLD
    BESTRUN(BESTJD)=HOLDD
    GO TO 50
   ELSE
    IF(FLAG.EQ.1)THEN
     BESTRUN(BESTJ)=HOLD
     GO TO 50
    ELSE

C SET TOTRUNS TO BEST AND
C GET FINAL BEST TIME
     TOTIME=0
     DO 100 J=1,JM1
       TOTRUN(J)=BESTRUN(J)
100  CONTINUE
    ENDIF
   ENDIF
   END
```

APPENDIX F. SIMULATED ANNEALING SUBROUTINE

```
      SUBROUTINE SIM(
     I         S,MU,TLIM
     U         ,TOTRUN,SEED)
      IMPLICIT NONE
C LIMITS ON MINE TYPES, SWEEP TYPES, MINES, RESOURCES, TRACKS
      INTEGER*4 O1,O2,O4,O5
      PARAMETER(O1=10,O2=9,O4=8,O5=40)
      REAL*4 A(O1,O2),B(O1,O2),AF(O1,O2),WIDTH,TR,TLIM
     +    ,BF(O1,O2),HOUR(O2),SIG(O2),COST,HT,TMPT
     +    ,ACT(O1),TMPC,VAL(O4),TOTIME,S(O1,O2),PNLTY
     +    ,CHAMP,R,SIZEFAC,MINPER,MCNT,TEMP,TEMP1,ALFA,TMP
     +    ,DELTA,SIGN,MPROB,PERACPT,OVERTIME,MU(O1,O2)
      CHARACTER NM(O1)*7,NS(O2)*8,NR(O4)*7
      INTEGER*4 IMAX,JMAX,KMAX,JM1,SCNT,MAXINT
     + ,IND(O2),SEQ(O2),H(O4,O2),J,TOTRUN(O2),TRIAL
     + ,L,CNT,N,TRACK,PLAN,DOWNJ(O2),SEED,PICKJ,U(O5)
     + ,CHAMPRUN(O2),TMPRUN(O2),FROZ,K
      COMMON A,B,AF,BF,HOUR,ACT,WIDTH,TR,SIG,VAL,HT
     +     ,SEQ,IMAX,JMAX,KMAX,IND,H
     +     ,NM,NS,NR
      DATA MAXINT /2147483647/
      JM1=JMAX-1

C GET INITIAL TIME AND COST
C AND SET PRESENT AND CHAMPION VALUES
      TMPT=0.
      DO 10 J=1,JM1
        TMPRUN(J)=TOTRUN(J)
        CHAMPRUN(J)=TOTRUN(J)
        TMPT=TMPT+TMPRUN(J)*HOUR(J)
10    CONTINUE
      TMPRUN(JMAX)=1
      CHAMPRUN(JMAX)=1

      CALL NF1(TOTRUN,S,MU,COST)

      TMPC=COST
      CHAMP=COST

C +++++++REMOVE COMMENT TO WRITE INTIAL PLAN TO SCREEN++++++
C     WRITE(6,*)'SIM ANNEAL INITIAL PLAN'
C     WRITE(6,*)'TIME = ',TMPT
C     WRITE(6,*)(TMPRUN(J),J=1,JM1)
C     WRITE(6,*)'COST = ',TMPC
C +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

C SIMULATED ANNEALING
C SET VARIOUS PARAMETERS

C COOLING RATIO: DETERMINES THE RATE AT WHICH
C THE CHANCE OF MOVING TO A WORSE PLAN DECREASES
      R=0.95

C FROZEN: A COUNTER IS INCREMENTED CONDITIONALLY
C EACH TIME A TEMPERATURE RUN IS COMPLETED. WHEN
```

```
C THE COUNTER EQUALS FROZEN THE ANNEALING PROCESS
C TERMINATES.
   FROZ=5
   SCNT=0

C MINPERCENT: AT COMPLETION OF A TEMPERATURE
C IF THE PERCENTAGE OF ACCEPTED MOVES
C IS <= MINPERCENT AND NO NEW CHAMPION PLAN
C WAS FOUND THEN AN INCREMENTAL MOVE TOWARD
C FROZEN OCCURS
   MINPER=3.
   MCNT=0.

C SIZEFACTOR: THIS IS MULTIPLIED
C BY THE SIZE OF THE NEIGHBORHOOD TO DETERMINE THE
C NUMBER OF MOVES (NEW PLANS) THAT WILL BE PROPOSED AT
C CURRENT TEMPERATURE. SIZEFAC*N=L (TEMP. LENGTH).
   SIZEFAC=25.

C TEMPERATURE: THIS IS THE STARTING TEMPERATURE WHICH
C IS SUBSEQUENTLY REDUCED AT THE AND OF EACH (TEMPERATURE
C LENGTH) NUMBER OF PROPOSED MOVES. (INITIALLY SET AS
C THE SUM OF ALL VALUES OF SWEEP TYPES AND HVU
   TEMP=0.
   DO 15 K=1,KMAX
     TEMP=TEMP+VAL(K)
15  CONTINUE
   TEMP=TEMP+VAL(KMAX+1)

C ALFA: THIS IS A PENALTY FACTOR APPLIED TO COST OF PLANS
C THAT ARE INFEASIBLE. THE AMOUNT OF INFEASABILITY (OVERTIME)
C SQUARED TIMES ALFA IS ADDED TO THE COST.
   ALFA=.0005

   TRIAL=0

C BEGIN PROCESS AND CONTINUE WHILE NOT FROZEN
20  IF(SCNT.LT.FROZ)THEN

C ++++++++REMOVE COMMENT TO WATCH TEMPERATURE DECREASE++++++
C   WRITE(6,*) 'TEMP = ', TEMP
C ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

C RESET TEMPORARY AND TOTAL RUNS TO CHAMP
C FOR START OF NEW TRIAL
     TMPT=0
     TMPC=CHAMP
     DO 30 J=1,JM1
       TOTRUN(J)=CHAMPRUN(J)
       TMPRUN(J)=CHAMPRUN(J)
       TMPT=TMPT+TMPRUN(J)*HOUR(J)
30   CONTINUE

C DETERMINE CURRENT NEIGH (N) SIZE AND COMPUTE TEMP LENGTH (L)
     N=JM1*2.
     L=INT(N*SIZEFAC)

     MCNT=0.
     TRACK=0
```

```
C BEGIN BEGIN BEGIN RUN AT CURRENT TEMPERATURE
    DO 60 PLAN=1,L

C DETERMINE NEW NEIGH SIZE AND STORE J's FOR WHICH
C REDUCING THE NUMBER OF RUNS IS FEASIBLE
      CNT=0
      N=0
      DO 40 J=1,JM1
       IF(TOTRUN(J).GT.0)THEN
        CNT=CNT+1
        DOWNJ(CNT)=J
       ENDIF
40   CONTINUE
      N=JM1+CNT

C GET A RANDOM NEIGHBOR OF CURRENT PLAN BY GENERATING
C A RANDOM NUMBER BETWEEN 1 AND N
      CALL RAND(1,U,SEED)
      TMP=1.0*MAXINT
      TMP=(U(1)-1)/TMP
      PICKJ=1+(TMP*N)

      IF(PICKJ.LE.JM1)THEN
       TOTRUN(PICKJ)=TOTRUN(PICKJ)+1
       SIGN=1.0
      ELSE
       PICKJ=PICKJ-JM1
       PICKJ=DOWNJ(PICKJ)
       TOTRUN(PICKJ)=TOTRUN(PICKJ)-1
       SIGN=-1.0
      ENDIF

C DETERMINE IF NEW PLAN IS FEASIBLE, AND IF
C NOT COMPUTE THE PENALTY TO APPLY TO ITS COST
      TOTIME=TMPT+(SIGN*HOUR(PICKJ))
      IF(TOTIME.GT.TLIM)THEN
       OVERTIME=TOTIME-TLIM
       PNLTY=ALFA*OVERTIME*OVERTIME
      ELSE
       PNLTY=0.
      ENDIF

C GET COST WITH NEW PLAN
      CALL NF1(TOTRUN,S,MU,COST)
      COST=COST+PNLTY
      DELTA=COST-TMPC

C WHEN DELTA VERY NEAR ZERO SET TO CONSTANT
      IF(DELTA.LT.0.0001)DELTA=.0001

C CHECK IF PROPOSED PLAN'S COST IS BETTER THEN CURRENT
C PLAN'S COST AND IF SO ACCEPT IT.
      IF(COST.LE.TMPC)THEN
       TMPC=COST
       TMPRUN(PICKJ)=TOTRUN(PICKJ)
       TMPT=TOTIME
       MCNT=MCNT+1

C WHEN PROPOSED PLAN'S COST IS WORSE ACCEPT IT
```

```
C WITH SOME PROBABILITY
      ELSE
       DELTA=DELTA*1000.
       TEMP1=TEMP*1000.
       MPROB=EXP(-1.0*DELTA/TEMP1)
       MPROB=MPROB*1000.
       CALL RAND(1,U,SEED)
       TMP=1.0*MAXINT
       TMP=(U(1)-1)/TMP
       TMP=TMP*1000.
       IF(TMP.LE.MPROB)THEN
        TMPC=COST
        TMPRUN(PICKJ)=TOTRUN(PICKJ)
        TMPT=TOTIME
        MCNT=MCNT+1
       ENDIF
      ENDIF

C WHEN PROPOSED PLAN IS ACCEPTED AND FEASIBLE
C COMPARE TO BEST PLAN AND UPDATE BEST PLAN IF
C PROPOSED PLAN IS BETTER
      IF(TMPRUN(PICKJ).EQ.TOTRUN(PICKJ))THEN
       IF(TMPT.LE.TLIM)THEN
        IF(TMPC.LT.CHAMP)THEN
         CHAMP=TMPC
         DO 50 J=1,JM1
          CHAMPRUN(J)=TMPRUN(J)
50  CONTINUE
         TRACK=1
        ENDIF
       ENDIF
      ENDIF

C RESET RUNS AND TIME IN CASE PROPOSED PLAN
C WAS NOT ACCEPTED
      TOTRUN(PICKJ)=TMPRUN(PICKJ)
      TOTIME=TMPT
60  CONTINUE

C COMPLETED COMPLETED COMPLETED CURRENT TEMPERATURE
    TRIAL=TRIAL+1

C ++++++REMOVE COMMENTS TO WRITE TRIAL RESULTS TO SCREEN++++++
C    WRITE(6,*) 'TRIAL = ', TRIAL
C    WRITE(6,*) (TMPRUN(J),J=1,JMAX)
C    WRITE(6,*) 'CURRENT PLANs TIME = ',TMPT
C    WRITE(6,*) 'CURRENT PLANs COST = ',TMPC
C    WRITE(6,*) 'NUMBER OF ACCEPTED PLANS = ',MCNT
C ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

C CALCULATE PERCENT OF ACCEPTED PLANS
    PERACPT=100*MCNT/L

C CHECK IF THIS TEMPERATURE FOUND A NEW BEST PLAN
C AND IF SO THEN RESET COUNTER FOR PROCESS TO 0
    IF(TRACK.EQ.1)THEN
     SCNT=0

C +++++++++REMOVE COMMENT TO INDICATE IMPROVEMENT FOUND++++++++
```

```
      WRITE(6,*)'**IMPROVING**'
C +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

C OTHERWISE IF THE PERCENTAGE OF ACCEPTED PLANS
C IS <= MINPER INCREMENT PROCESS COUNTER
     ELSEIF(PERACPT.LE.MINPER)THEN
        SCNT=SCNT+1
     ELSE
     ENDIF

C COOL TEMPERATURE TO NEXT LEVEL
     TEMP=R*TEMP

     GO TO 20
    ENDIF

C ANNEALING RUN COMPLETED

C CHECK FINAL ACCEPTED PLAN FROM ANNEALING
C RUN FOR FEASIBILITY. IF INFEASIBLE INCREASE
C PENALTY AND DO ANOTHER ANNEALING RUN
    IF(TMPT.GT.TLIM)THEN
     SCNT=FROZ-1
     ALFA=ALFA*2.
     GO TO 20
    ENDIF

C SET TOTRUN TO CHAMP
    DO 100 J=1,JM1
      TOTRUN(J)=CHAMPRUN(J)
100   CONTINUE
    END
```

LIST OF REFERENCES

1. Washburn, Alan R., *MIXER: A TDA for Mixed Minefield Clearance*, Monterey, CA, September 1995.

2. Boorda, Jeremey M., *Mine Countermeasures - An Integral Part of Our Strategy and Our Forces*, Wash. D.C., December 1995.

3. Washburn, Alan R., *Mine Warfare Models*, Monterey, CA, October 1995.

4. McCurdy, Michael L., *A Cognitive Planning AID for Naval Minesweeping Operations*, Camp H. M. Smith, HI, April 1988.

5. Washburn, Alan R., *KATZ Distributions, with Applications to Mine field Clearance*, Monterey, CA, March 1996.

6. Loukides, Mike, *UNIX for FORTRAN Programmers*, O'Reilly & Associates, Inc., Sebastopol, CA, 1990.

7. Johnson, David S. And others, "Optimization by Simulated Annealing: An experimental evaluation; Part 1, Graph Partitioning", *Operations Research*, Vol. 37, No. 6, pp. 865-892, November - December 1989.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Technical Information Center 8725 John J. Kingman Road., Ste 0944 Ft. Belvoir, VA 22060-6218	2
2.	Dudley Knox Library Naval Postgraduate School 411 Dyer Rd. Monterey, CA 93943-5101	2
3.	Captain Frank C. Petho, USN (Code OR) Chairman, Operations Research Dept. Naval Postgraduate School Monterey, CA 93943-5000	1
4.	Professor Allan R. Washburn (Code OR/WS) Department of Operations Research Naval Postgraduate School Monterey, CA 93943-5000	1
5.	Professor James N. Eagle (Code OR/ER) Department of Operations Research Naval Postgraduate School Monterey, CA 93943-5000	1
6.	COMINEWARCOM (N02R) 325 Fifth Street SE Corpus Cristi, TX 78419	1
7.	Coastal Systems Station 6703 W. Highway 98 Panama City, FL 2407-7001	1
8.	Center for Naval Analyses 4401 Ford Avenue P.O. Box 16268 Alexandria, VA 22302-0268	1
9.	Office of Naval Research (322TE) 800 North Quincy Street Arlington, VA 22217-55660	1
10.	Daniel H, Wagner, Assoc. 2 Eaton Street, Suite 500 Hampton, VA 23669	1

11. David H. Romberger, Esq. 1
512 71st Street
Holmes Beach, FL 34217

12. LT David D. Romberger, USN 3
512 71st Street
Holmes Beach, FL 34217